Semantic Mapping for Indoor Robotics

Krishnananda Prabhu Sivananda

Thesis submitted for examination for the degree of Master of Science in Technology under the dual degree program of EIT Digital Master School.
Espoo 21.12.2020

Supervisors

Prof. Ville Kyrki

Prof. Sahin Albayrak

Advisors

Dr. Francesco Verdoja

Dr. Orhan Can Görür

Aalto University
P.O.BOX 11000, 00076 AALTO
www.aalto.fi

Technische Universität Berlin
Franklinstrasse 28-29, 10587 Berlin
www.av.tu-berlin.de

Hereby I declare that I wrote this thesis myself with the help of no more than the mentioned literature and auxiliary means.

Espoo, 21.12.2020

. . . . . .

(Signature [Krishnananda Prabhu Sivananda])

Technische Universität Berlin

Abstract of the master's thesis

Aalto University
School of Electrical Engineering

Author Krishnananda Prabhu Sivananda

Title Semantic Mapping for Indoor Robotics

Degree programme Master's Programme in ICT Innovation (EIT Digital Master School)

Major Autonomous Systems **Code of major** ELEC3055

Supervisors Prof. Ville Kyrki, Prof. Sahin Albayrak

Advisors Dr. Francesco Verdoja, Dr. Orhan Can Görür

Date 21.12.2020 **Number of pages** 69 **Language** English

Abstract

Advancements in autonomous robotics research have reached a stage where mobile robots are being deployed in social environments such as offices, malls, and households. Mobile robots deployed in such crowded environments often need to interact with objects to perform a task or to navigate through the environment to reach their destination. It is imperative that the robot has knowledge about the semantics and the complete geometry of the object it needs to interact with, because interaction with incomplete information can cause safety risks to the object or the surroundings. The goal of this thesis was to develop a method that generates semantic maps with complete geometric information of objects in the environment using color and depth sensors. To this end, a pipeline was implemented to construct a mesh representation of the environment with partial object representations in the mesh replaced with similar synthetic models. The pipeline also explores the viability of a deep learning based approach to complete the missing regions in the object representations. Tests were conducted to evaluate the pipeline on an office environment simulated in Gazebo and on a real environment using a Care-O-bot robot. The pipeline performs well when the pace of the robot navigating the environment is low. The comparison between the deep learning based approach and the synthetic model matching approach favors the latter for better quality output. Furthermore, the analysis of results indicated that the 2D image segmentation method used limits the performance of the pipeline as a whole. Future directions to expand the work can include expansion of the database to accommodate multiple object classes, using an alternate method for image segmentation, and exploring the possibility of applying the output generated by the pipeline for 'Interactive Navigation' research.

Keywords geometric reconstruction, semantic reconstruction, object segmentation, computer vision , deep learning , point clouds , mesh

Preface

I want to thank Prof. Ville Kyrki and my advisor Dr. Francesco Verdoja for the incredible support and guidance throughout the thesis. I want to convey my Gratitude to Prof. Sahin Albayrak and Dr. Orhan Can Görür for providing the opportunity to work on this thesis.

I would also like to thank Aino Roms and Chi-Thanh Christopher Nguyen for their help and guidance with the administration. Additionally, I also would like to thank my family and friends for their emotional support and care.

Espoo, 21.12.2020

Krishnananda Prabhu Sivananda

Contents

Symbols and abbreviations

Abbreviations

AMR	Autonomous Mobile Robot
CNN	Convolutional Neural Network
DCNN	Deep Convolutional Neural Network
DoN	Difference of Normals
FCN	Fully Connected Network
FPN	Feature Pyramid Network
GT	Ground Truth
ICP	Interactive Closest Point
PCN	Point Completion Network
RoI	Region of Interest
ROS	Robot Operating System
SDF	Signed Distance Field
SLAM	Simultaneous Localization and Mapping
TSDF	Truncated Signed Distance Field

1 Introduction

Early applications of robots were limited to assist or replace humans in performing repetitive and dangerous tasks. These traditional robots were mostly stationary and confined to industrial applications such as welding, assembly, picking, packaging. In 1948, William Grey Walter introduced his two prototype Autonomous Mobile Robots (AMR), Elmer and Elsie, for research purposes in the field of neurophysiology. They were equipped with a light sensor and a bump sensor with a vacuum tube to simulate connected neurons. The subsequent versions of the robots included additional sensors, enabling them to learn about the surroundings through navigation and to find the charging station autonomously through exploration [1]. The demonstration of their capabilities attracted the scientific community towards this field to come up with ways to use robots in more complex task settings and environments. Parallel advancements in Artificial Intelligence and Machine Learning helped researchers to develop more sophisticated robotic systems. In these modern times, mobile robots are being deployed everywhere from households to space stations for various applications. However, growing complexity in robotic systems poses various technical challenges as well as social implications to address.

A key challenge with AMR is navigation within indoor environments. In a traditional sense, it is solving the problem of autonomously moving the robot from one location to another. A navigation system requires an abstract understanding of the environment to plan its path through the environment. Based on the level of abstraction of the environment representation, navigation systems can be classified into three main domains: topological, geometric, and semantic [2].

In the topological approach, the environment representation is characterized by edges and nodes where aspects of the environment like rooms or corridors are considered nodes, and the relations between them are marked as edges. Such representations are compact as only distinctive places are encoded and computationally less demanding as precise localization is not required compared to metric maps [3].

The geometric approach is a well known and classical approach where the environment is represented in the form of metric maps like Occupancy grid maps, which encode free space and occupied space in a metric form. Simultaneous Localization and Mapping (SLAM) algorithms are commonly used methods to generate such metric representation [4]. In 3D, sensors like RGBD cameras are used to fuse the incoming 3D geometric information to build coherent maps that visually look similar to the actual environment.

The semantic approach is a fairly new and trending field of research where the environment model also carries the interpretation of the underlying geometric data. This allows providing robots cognitive architectures that can have behavioural mechanisms based on human psychology. The addition of semantic knowledge allows the robot to perceive the environment similar to humans aiding the navigation to be more robust, and efficient. Also, it enables the robot to achieve complex manipulation tasks and facilitates human-robot interactions. In 3D, current semantic mapping techniques combine deep learning based image segmentation solutions with 3D geometric mapping methods to encode the semantic information of the underlying

geometry.

1.1 Problem statement

In classical robot navigation, the robot finds a collision-free path avoiding all the obstacles to reach its destination. This interpretation works well in domains such as factories, warehouses, and outdoor scenarios. However, the low manufacturing costs of robots have aided their wide deployment in unstructured and complex environments such as offices, households, and shopping malls. With a solution based on the classical approach, when encountered with an unregistered obstacle the robot either comes to a halt to replan its trajectory or takes extreme evasive actions to avoid a possible collision. Such situations are frequent in a cluttered environment like an office or a household where there can arise situations when every path of the robot is expected to be obstructed by human interference or there is significant uncertainty in human trajectory estimations. This is called the freezing problem [5]. Such behaviour poses safety risks or inefficient usage of the robot. To solve the freezing problem, the robot has to turn to movable obstacles such as tables and chairs to interact with them to create a navigable path towards its destination. The research field that deals with robot navigation in congested spaces with the aid of manipulating movable obstacles is known as Interactive Navigation.

In order to interact with any object in the surrounding the robot needs to know the semantics of the object, geometry, and the pose which the classical methods fail to provide. Even though a 3D semantic map provides the necessary information, it is quite far from being perfect. A major drawback of existing semantic mapping methods is that the underlying geometry may be incomplete as it depends on factors such as occlusions, environmental limitations while perceiving the scene during the mapping process. A robot interacting with an object without knowing its complete extent can pose safety risks either to the object or the surroundings. This thesis addresses the problem of how to form a complete representation of objects from the map generated with existing methods.

1.2 Objective

Manually programming the robot to interact with different kinds of objects is impractical. A logical approach is to employ some learning strategies that enable the robot to learn and adapt through interaction with different kinds of objects. However, such an approach to apply on real robots will be expensive and interaction with the environment poses safety risks. Also, the capability of the current robots to learn from the actual environment is quite limited. Hence a sensible solution is to apply and test the learning strategies in a simulated environment with a physics engine. However, to our knowledge the availability of such a simulation environment with textures close to the real-world, and with interactable objects is limited to iGibson [6]. iGibson is a recently published work that has a similar objective to ours but is different in approach. This work validates our assumption of the necessity of providing better simulation environments with interactable objects for Interactive

Navigation research.

The objective is to generate a map of the environment that holds the complete information of objects which is required by a robot to safely interact with it. The generated map can be then used in a simulation platform like Gazebo to train the robot to interact with the objects in the map.

The main question to address to realize the objective is how to determine the complete geometry of an object when its partial geometry is known. There exist methods within Computer Vision (CV) research, that are deep learning based methods that attempt to complete the geometry of partially seen objects. However, the use of these methods in robotics is yet to be fully explored. The black-box nature of deep learning methods makes the output provided by the network unpredictable which is not desired when it comes to its application in robotics for purposes such as navigation or manipulation. A more deterministic approach would be to replace the partial objects altogether with a similar synthetic model from a known database. Since a deterministic approach always provides a completely noiseless match to the partial object, developing such a method was the primary focus for this thesis. The thesis also evaluates and compares the performance of a learning based approach for shape completion against our proposed approach.

1.3 Overview of the solution

The end result of this thesis is a pipeline that generates a semantic-geometric representation of an environment with a complete representation of objects obtained by replacing the partial objects with the corresponding match from the known database. Figure 1 depicts a high-level overview of the pipeline. The idea is to perform geometric mapping and semantic mapping of the same environment one after the other, use semantically annotated 3D geometry of objects to find a match from a known database of synthetic models, remove the objects from the geometric map and replace them with the corresponding synthetic model matching the pose of the actual object.

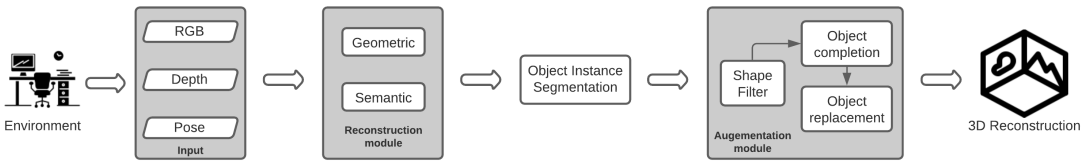


Figure 1: Overview of the pipeline.

An RGBD camera mounted on a robot acts as the input source of the pipeline. The robot navigates through the environment capturing and transmitting live RGB and depth images to the reconstruction modules. The geometric and semantic reconstruction modules produce a 3D geometric reconstruction and a corresponding semantically annotated geometric reconstruction of the scene respectively. The geometric reconstruction may have a lot of missing information especially with

regards to the extent of the objects which may have occurred due to occlusions, the inability of the robot to capture a complete view of the object, or other environmental limitations. The subsequent stages in the pipeline are intended to mitigate the extent of the missing information of objects in the current reconstruction. In the further stages, the reconstructed semantic scene is broken down and separated into clusters, each cluster representing an object. In the final stage, the complete representation of each object is determined. The object representations in the geometric scene are removed and their completed versions are inserted back into the scene with the original pose of the removed objects.

To summarize, the contributions of this thesis are as follows:

- A pipeline designed and implemented to generate a 3D mesh representation of an environment that contains synthetic models that are similar to the actual objects in the real environment.
- The implementation and evaluation of a novel method to compare and match partial observations of 3D object instances against a database of synthetic models and to replace the partial 3D mesh with the matched model in the reconstructed 3D mesh of the environment.
- The evaluation and comparison of a learning-based approach to complete partial observations of 3D object instances against our proposed method.

1.4 Structure of the thesis

The rest of the thesis is organized as follows:

Chapter 2 provides the background information and the current scientific research on the modules that constitute the pipeline. Chapter 3 describes the design choices and the implementation details that went into developing each module in the pipeline. Chapter 4 presents detailed analysis and conclusions on different parts of the implemented pipeline based on the experiments conducted in a simulated as well as a real environment. Finally, Chapter 5 provides a summary of all the findings and proposes future work for improving the pipeline, and to explore possible applications of the output of the pipeline.

2 Background and related work

This chapter presents the background used to develop the pipeline and introduces related work published by other researchers. In Section 2.1, different ways of representing 3D objects are compared. Section 2.2 introduces research in the area of 2D pixel-level semantic segmentation. Section 2.3 explains different methods for scene reconstruction in 3D. Section 2.4 describes object completion methods from partial observations in 3D.

2.1 Representing objects in 3D

Raw 3D data gathered from sensors such as RGBD cameras, and Laser scanners come in different forms that vary in both structure and properties. Depending on the purpose, the raw data can be processed and stored as different types of representation. The most popular and widely used representations and their benefits are discussed briefly in Table 1.

2.2 Instance-aware semantic segmentation methods

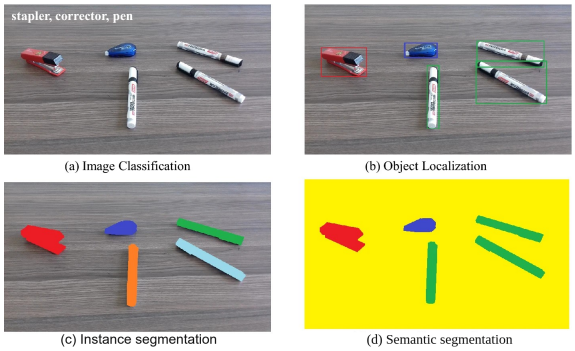


Figure 2: Approches for object recogniton. [8]

Object detection has been a key area of research in computer vision for decades. The goal of object detection is to locate instances of objects from known categories in an image and return their spatial location and extent in a particular form like a bounding box. Since their emergence, deep learning techniques, especially Deep Convolutional Neural Network (DCNN), have become an inherent part of all state of the art methods for this task because of their ability to learn complex, subtle, and abstract features directly from raw images. Semantic segmentation is a parallel area of research where the objective is to assign a label to each pixel according to the respective object class. However, this approach does not distinguish between different instances of the same object class. Hence, instance-aware semantic segmentation can be considered to be a unified task of solving the problem of object detection along with semantic segmentation. Figure 2 gives a comprehensive idea of different approaches for object recognition.

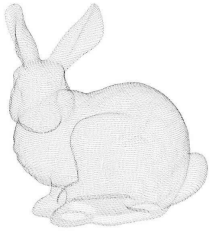
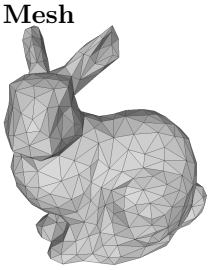
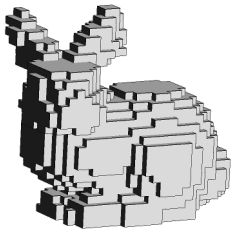
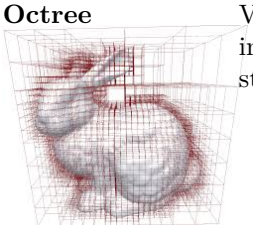
Type	Content	Benefits	Drawbacks
Pointcloud 	Unstructured points in a 3D coordinate system	• Fast rendering. • Exact representation. • Fast transformations • No discretization of data • Easy to visualize	• Unbounded memory consumption. • No direct representation of free/unknown space. • No connectivity information
Mesh 	Vertices connected by edges making faces of a polygonal shape	• Fast rendering • Exact representation • Easy to visualize	• High memory requirement • Approximated curved surfaces
Voxel 	Collection of non-overlapping cubes, also called voxels	• Volumetric representation • Possible to update probabilistically • Possible to encode viewpoint information • Trivial boolean operations are possible	• Cubically growing memory • Discretization errors • Does not preserve geometry • Not suitable to represent high-res data • Transformations are costly
Octree 	Varying sized voxels in tree-based data structure	• Volumetric representation • Possible to update probabilistically. • Efficient memory usage	• Inaccurate representation • Complicated implementation

Table 1: Comparison of different 3D data representations. [7]

The detection frameworks employed to perform any of the above tasks can be broadly classified into region-based (two-stage) and single-stage frameworks [9].

2.2.1 Two-stage frameworks

In a region-based (Two-stage) framework, first, category-independent region proposals are generated and then classified into respective object classes. Though the detection quality is high, the computational burden and low output frame rate are serious drawbacks of these types of frameworks. Mask R-CNN [10] based on Faster R-CNN [11] follows a top-down approach where box prediction followed by segmentation is performed. It is an efficient method that generates high-quality segmentation masks along with box regression and object classification running at 5 fps. The improved performance is achieved with the addition of a parallel branch to the Feature Pyramid Network (FPN) based architecture of Faster R-CNN. The parallel branch consists of Fully Connected Network (FCN) layers that predict binary masks from each of the Region of Interest (RoI) proposals which increase the accuracy of segmentation masks without the significant computational overhead. Another advantage of this approach is that it is easily extensible for human pose estimation. An alternate approach that has comparable performance to Mask R-CNN which is also based on Faster R-CNN is MaskLab [12] which does foreground and background segmentation inside each box prediction by a combination of direction prediction and semantic segmentation. The semantic segmentation model assists in background removal, and direction prediction which is an estimation of each pixel's direction towards its respective center assists in separating instances of the same object class. Some of the more recent approaches like PANet [13] , MegDet [14] , Hybrid Task Cascade (HTC) [15] which performed well in various detection challenges are built on top of the Mask R-CNN but brings only minor improvements in terms of accuracy and computational efficiency.

2.2.2 Single-stage frameworks

Single-stage frameworks consist of a single feed-forward network without region proposal which can be optimized end-to-end. The purpose of single-stage frameworks is to accelerate the generation of results; they achieve it by eliminating the second stage and compensate for lower performance in other ways. YOLO [16] and SSD [17] are prime examples for such methods. However, the same approach is difficult to apply on instance segmentation because all the high performing two-stage methods use the features inside bounding box regions as an input to the mask predictor, which is a sequential process. Parallelizing these steps for one stage segmentation is challenging. YOLACT [18] tries to address this issue by adding a parallel module that produces instance masks as a linear combination of prototype masks and their coefficients. It runs real-time achieving 33.5 fps but falls behind two-stage methods in terms of accuracy. As an alternate approach that takes on a new perspective, TensorMask [19] uses sliding window technique that generates predictions of bounding box with the aid of dense and regularly spaced grids. The key idea proposed in this method is the use of a 4D tensor representation of masks over a spatial domain,

where the tensor dimensions consist of object position and relative mask position. This method shows results comparable to that of Mask R-CNN [19].

2.3 3D Scene reconstruction methods

3D reconstruction of a scene has a wide variety of applications in the fields such as Augmented reality (AR), Virtual Reality (VR), medical imaging, and robotics. The availability of cheap RGBD cameras like Kinect has resulted in profound advances in developing 3D reconstruction methods. The RGBD data from the sensor is sparse and incomplete, hence multiple RGBD images need to be globally registered and the aligned images have to be fused into a volumetric or pointset representation. Truncated Signed Distance Function (TSDF) is a volumetric concept of representing 3D surfaces where each voxel in the 3D grid is mapped to the nearest surface. KinectFusion [20] is a well-known method that popularised TSDF based reconstruction which is capable of real-time reconstruction of small environments. The main drawback of pointset representation is that it is not possible to model the empty and occupied space. Hence most of the methods which are developed for applications like robot navigation rely on volumetric representation. Based on the purpose of reconstruction the methods can be classified as either Geometric or Semantic [21].

2.3.1 Geometric reconstruction methods

The main drawback of KinectFusion is the use of a fixed-size voxel grid which requires the map size to be known and a large amount of memory. Multiple extensions have been proposed to address these shortcomings. Whelan *et al.* [22] proposed a dynamic fixed-size TSDF volume and converting the regions not under consideration into the triangular mesh. Steinbrucker *et al.* [23] use octree representation to dynamically allocate the volume only around the observed region to save memory and computation time. The algorithm is capable of running geometric fusion at real-time on a CPU approximately at 45 Hz but the incremental mesh update happens only at a frequency of 1 Hz. Nießner *et al.* [24] propose a spatial hashing scheme to compress space while allowing real-time access and updates of underlying surface data. This spatial hashing was faster than the hierarchical grid data structure used in other methods. However, it relies heavily on GPUs for real-time performance. Voxblox [25] focuses on improving memory efficiency and real-time performance on the CPU. The method builds incremental Euclidean Signed Distance Fields (ESDF) from TSDFs using the spatial hashing technique employed in Nießner *et al.*.

2.3.2 Semantic reconstruction methods

Robots get to act better when they also have the semantic information in the environment along with the geometry. Advancement in object detection and segmentation through deep learning techniques has pushed researchers to find ways to integrate semantics into the mapping methods. Now, there exist a plethora of methods which combine deep learning based object detection with 3D mapping methods to generate information-rich maps which the robot might require to perform complex tasks.

SemanticFusion [26] combines ElasticFusion [27] which is a real-time SLAM system and a CNN for object detection with a bayesian update scheme for semantic label integration. An average frame-rate of 25.3 Hz is reported with the aid of a GPU. Voxblox++ [28] is an extension of voxblox framework which performs geometric reconstruction. Semantic labels are generated by combining geometric segmentation of depth information through the difference of surface normal calculation and instance segmentation with Mask R-CNN. A data association strategy keeps track of labels to ensure global consistency. It operates at a frame rate of 1 Hz. Kimera-Semantics [29] is another approach that uses Voxblox framework with flexibility for choice of 2D instance segmentation algorithm. It runs at 10 Hz and shows accurate reconstruction when compared against Ground Truth (GT).

2.4 3D Object shape completion methods

A robot often encounters tasks that require interaction with objects, hence the robot must have both semantic and geometric information about the object that it needs to interact with. The information sources are common 3D sensors such as RGBD cameras, laser scanners, or multiview stereo systems. These sensor inputs tend to have incomplete information about the measurement area because of occlusions, sensor noise, or other environmental limitations. Robots interacting with objects without knowing their complete extent can pose safety risks to themselves, to a human, or the object itself. So it is imperative to provide complete information about the extent of the object for any robotic task involving its manipulation. The missing geometric information can be inferred by applying shape completion methods on 3D partial scans or by replacing the noisy and incomplete geometry with matching synthetic models.

2.4.1 Prediction methods

Early attempts in shape completions focused on filling small missing regions where the majority of the geometry is known. Surface reconstruction methods proposed by [30]–[32] attempt to fit geometric primitives through convex interpolation or by solving the Poisson equation to obtain a smooth surface to approximate the missing region. Alternatively, the works of [33], [34] propose reconstruction from oriented point clouds by interpreting them into a Poisson formulation to locally optimize for missing surfaces. The presence of symmetry and repeated geometric structures within an object is also a subject of research to formulate methods for reconstructing objects exploiting this information. Works like [35]–[37] try to extract symmetry information such as spherical, planar, or euclidean symmetry to reconstruct the missing part of 3D partial scan whereas, Pauly *et al.* [38] study the repeating occurrences of regular structures in an object and copy those to the unobserved regions to complete the extent of the object. All of the approaches above heavily rely on the percentage of the unobserved regions and the quality of the partial scans.

Recent methods mostly rely on deep learning techniques to predict the scan completion. The network architecture choice depends on the type of data representation

at hand. Popular representations include voxels, mesh, and point cloud.

Voxel-based

CNN-based architectures depend on convolution operation for feature learning which requires a structured grid and hence most of the methods use a voxelized representation of the scan input. Dai *et al.* [39] propose a data-driven approach where voxelized synthetic models and 3D partial scans are mapped in an embedded latent space representation. The method loosely follows the idea of autoencoders for learning efficient encoding. Sharma *et al.* [40] propose a similar data-driven approach, where a fully convolutional volumetric encoder is used to learn embedding of object shape from noisy data by estimating voxel occupancy grids. Han *et al.* [41] incorporate the global structure inferred by a Long Short-Term Memorized Context Fusion module (LSTM-CF) with an encoder-decoder network to generate a complete output. However, cubically growing memory of the voxel representation increases network complexity and computational overhead which severely limits the output resolution [42].

Mesh-based

Deep learning methods using mesh representations are scarce because of two reasons: learning irregular representations like mesh is challenging for neural networks, properties like shift-invariance, operations of the vector space, and the global parametrization system are absent because of their non-Euclidean nature [7]. Litany *et al.* [43] explore deformable meshes for shape representation. The method uses a variational autoencoder with graph convolution operation to learn the latent space. However, the method assumes all the shapes are in correspondence with a common reference shape, hence limiting its applicability.

Point cloud-based

The data acquisition from sensors like laser scanners, stereo, and RGBD cameras mostly favors point cloud format. Hence it is desirable to use such raw input directly for learning methods without any intermediate preprocessing. But properties of point clouds like irregularity, lack of structure, and unorderedness make it challenging to directly apply deep learning methods [44]. PointNet [45] is the first approach that addresses the above challenges and applies deep learning on unstructured raw point cloud data. PointNet is made up of symmetric functions whose outputs are permutation independent, multilayer perceptrons (MLP), and a max-pooling function. It achieves state-of-the-art performance on several benchmark datasets and has become the basis of many methods that were developed since then.

Point Completion Network (PCN) [46] is an encoder-decoder network based shape completion method that uses PointNet architecture for the encoder design to encode a feature vector. It is capable of generating high-resolution completion and shows generalization over unseen objects and real-world data. Point Fractal Network (PF-Net) [47] is capable of predicting the geometrical structure of the

missing regions in the input partial point cloud preserving the spatial arrangement of the input, unlike PCN where it predicts the overall shape from the incomplete point cloud. Chen *et al.* [48] analyzed that the networks trained on synthetic-partial and synthetic-complete data do not translate well to real data, hence proposes a method that learns to translate noisy and incomplete point clouds into complete pointsets. It used the Generative Adversarial Network (GAN) to learn a mapping from partial to complete representation. However, on the performance front, it falls behind PCN. TopNet [49] is a different approach wherein a novel tree decoder network is used to model and embed the defining structure on a point cloud. The network generates a structured point cloud without any assumptions over the topology of the underlying point set. The authors also claim a 30% improvement on shape completion task over the next best performing method which is PCN.

In general, these methods tend to produce noisy output which is not desired for the intended use cases of robotic applications. Moreover, they are unpredictable and their performance outside training data in a robotic context has not been properly explored.

2.4.2 Model matching methods

Another way to address object shape completion is to retrieve synthetic models from a database and align them to the input scan to get a more clean and better-looking scene representation. The low-level geometric features may differ significantly between the partial scan and the model while high-level geometry is similar. This limits the applicability of geometric feature descriptors such as Fast Point Feature Histograms (FPFH) [50], point-pair features [51] which makes it extremely difficult to match synthetic models to partial scans.

Scan2CAD [52] is an approach in this area. They introduced a dataset comprising reconstructed scans in ScanNet [53] and ShapeNet [54] along with pairwise keypoint correspondences. It also proposes a method based on CNN to learn joint embedding between real and synthetic 3D models to predict accurate correspondence heatmaps between model and scan. By performing energy minimization optimal 9DoF pose alignment between the model and the scan is determined. The model uses an entire 3D scan of a scene as input and tries to find the alignment of models of all the objects in the scene.

3 Pipeline design and implementation

This chapter describes the design choices and the implementation of different modules in the pipeline. Section 3.1 provides basic concepts of ROS, which was used in this thesis as middleware between different modules. Section 3.2 depicts a detailed overview of the entire pipeline describing the flow of information. From Section 3.3 detailed description of each module in the pipeline is presented starting with the reconstruction module. Section 3.4 describes the method used to cluster objects from the scene. Section 3.5 describes the last stage in the pipeline where incomplete objects in the scene are replaced with their completed versions.

3.1 Basic concepts of Robot Operating System

Some of the modules in the pipeline use ROS as the framework for information exchange; it is important to understand its basic concepts. Even though the name suggests that it is an operating system, it is quite far from it. ROS [55] is actually an open-source middleware software framework that serves as a platform for different software components to communicate with each other. Since it is an active open source community, a lot of researchers use this platform to write their robot software. ROS processes can be represented as nodes in a graph structure, connected by edges called Topics.

ROS Master

ROS Master is the main node that all the other nodes register to. ROS Master keeps track of all the registered nodes and facilitates information exchange between nodes through topics. The ROS Master does not handle the information directly; rather it sets up a peer-to-peer communication channel between nodes that wish to communicate with each other.

ROS nodes

ROS nodes are individual processes that perform different tasks. Each node has a unique identifier it registers itself to the Master with. Nodes can communicate with each other through messages. ROS messages are defined data structures that can carry different data types or message types. Topics are carriers of ROS messages with unique identifiers with which nodes can send or receive them. Message transfer between nodes follows a publisher-subscriber system. Nodes that want to send messages publish them to a topic and the nodes that wish to receive these messages subscribe to the respective topic. A node can also advertise services. Upon invoking a service request the node performs certain actions at the end of which it returns a result.

3.2 Detailed overview of the pipeline

Figure 3 depicts a detailed overview of all the software components used to form the pipeline. The pipeline can be split into two halves. The first half composes of modules for real-time reconstruction of a scene. It includes the geometric and semantic-geometric reconstruction modules. These reconstructions form the input to the second half of the pipeline. The second half consists of different modules working together to perform augmentation on the input to deliver the final geometric representation of the scene having objects with complete geometry. A detailed description of each of the individual software components used in the pipeline is discussed in the subsequent sections.

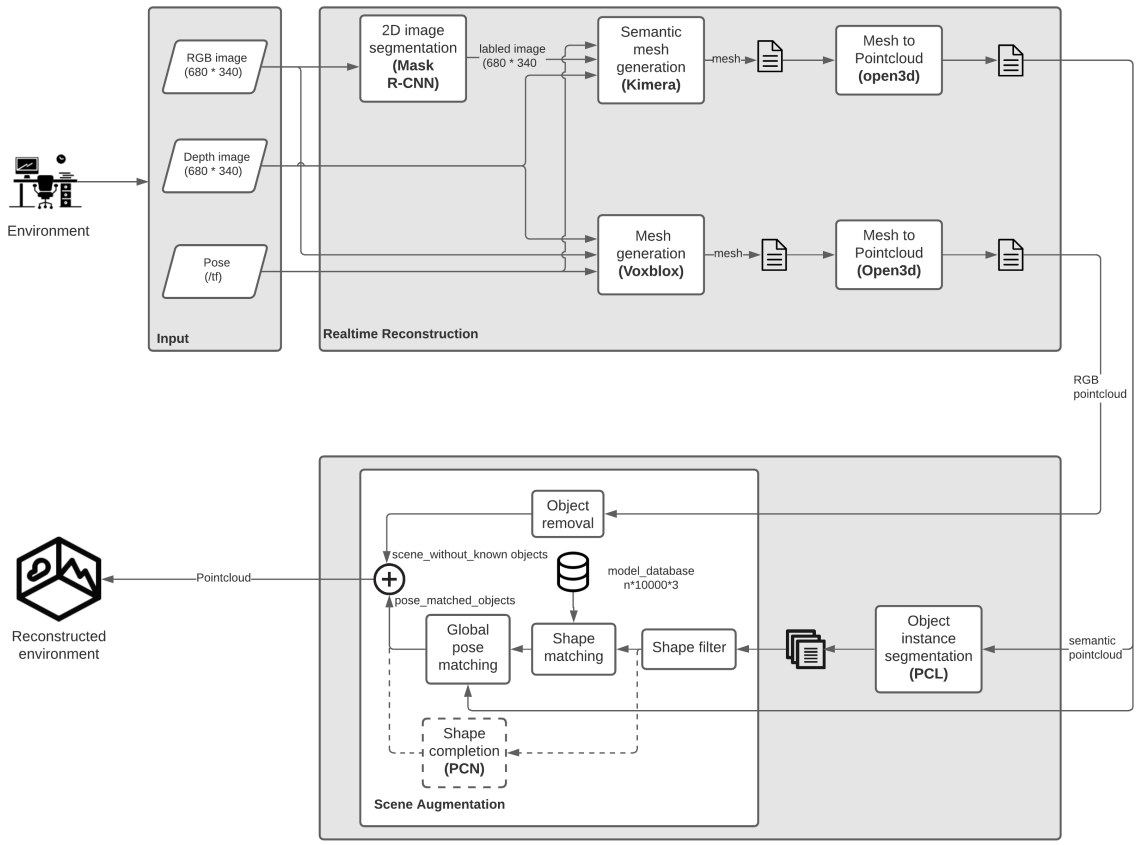


Figure 3: Detailed overview of the pipeline.

3.3 Reconstruction module

The purpose of the reconstruction module is to generate a geometric and semantic-geometric mesh online. Voxblox [25] was used for the geometric reconstruction. Kimera [29] was used for the semantic-geometric reconstruction with Mask R-CNN [10] providing the instance segmented images. The input to the reconstruction

module are live RGB images (640 * 480) and depth images (640 * 480), camera parameters, and the pose information provided as ROS topics.

3.3.1 Geometric reconstruction

The geometric reconstruction component is in charge of constructing colored geometric mesh of the scene in real-time. The requirements were that the reconstructed mesh should be of good resolution with textures and the mesh should be accessible offline. The component was implemented as a ROS node. Referring to section 2.3.1, the majority of the existing methods require heavy memory consumption and powerful GPU's to perform reconstruction in high resolution. Voxblox was the first method that was capable to run with memory efficiency solely using CPU resources which performed scene reconstruction with high resolution, hence was chosen for this task. Its following properties were of added advantage.

- Flexibility in providing pose information through different ways
- Extensive parametrization to tailor performance based on requirements
- Possible to save constructed mesh
- ROS implementation is available
- Detailed documentation

The incoming sensor data is integrated into a TSDF by grouped raycasting technique. Raycasting is a method by which a ray cast between the optical center of the camera and the observed points is used to update the voxels based on the truncation distance defined between the two. Additionally, instead of performing raycasting on all the observed points, they are projected onto a voxel grid and the points mapping to the same voxel are grouped together. The weighted mean of all points per voxel is then used to perform raycasting to the reduced set of points to increase the speed of reconstruction without compromising on quality. The mesh is constructed on demand in an incremental process on updated voxels using the marching cubes [56] algorithm. The library is also capable of producing an ESDF representation from the TSDF layer for the purpose of motion planning which is out of scope for our intended use. Figure 4 depicts the system architecture of the voxblox library with different layers for each type of representation.

Implementation details

A custom launch file in ROS was created to accommodate for different sensor settings and performance requirements. Since the emphasis of this module is on mesh quality all the mentioned parameters were set so as to produce mesh with high resolution which will impact computation time. The TSDF voxel size was set at 0.02 m. Integration of TSDF voxels was performed by ray bundling such that points lying in the same voxel are merged to increase integration speed.

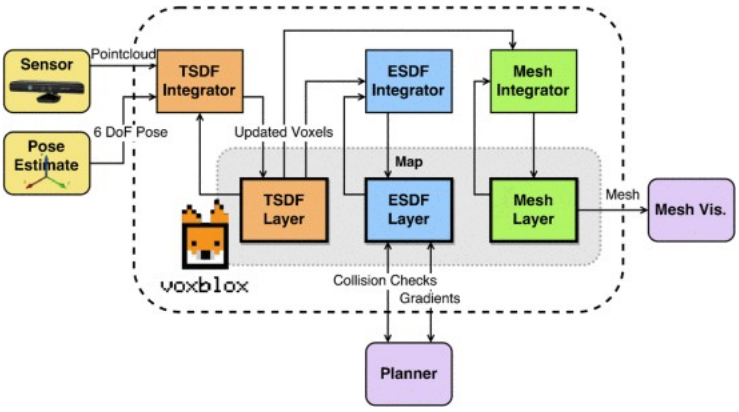
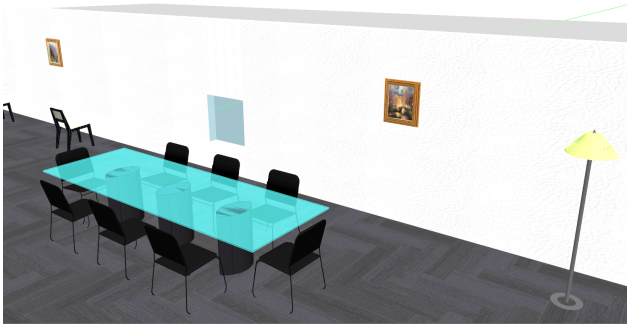
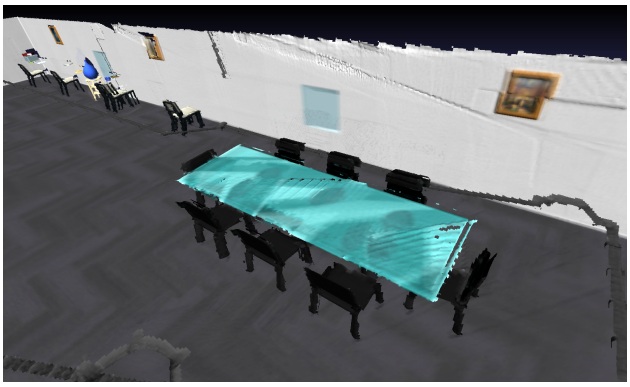


Figure 4: System architecture of voxblox. [25]



(a) Simulation.



(b) Geometric reconstruction.

Figure 5: Example output of Voxblox in simulation.

Upon invoking the ROS service call 'generate_mesh', the TSDF layer that is updated till that point in time will be integrated and the mesh will be saved in a .ply format. Figure 5 shows an example output generated by the Voxblox library in a simulation setup.

3.3.2 Instance segmentation

The purpose of instance segmentation is to identify and label all the individual objects in each image frame with a unique mask for each object. The algorithm should be able to generate instance masks close to the object boundaries and should be able to distinguish between multiple instances of the same object class. Additionally, detection accuracy should be reasonable. The component was implemented as a ROS node. Mask R-CNN was chosen for this task due to the following reasons:

- State of the art method for object instance segmentation
- Network weights trained on COCO dataset [57] is available
- ROS implementation is available

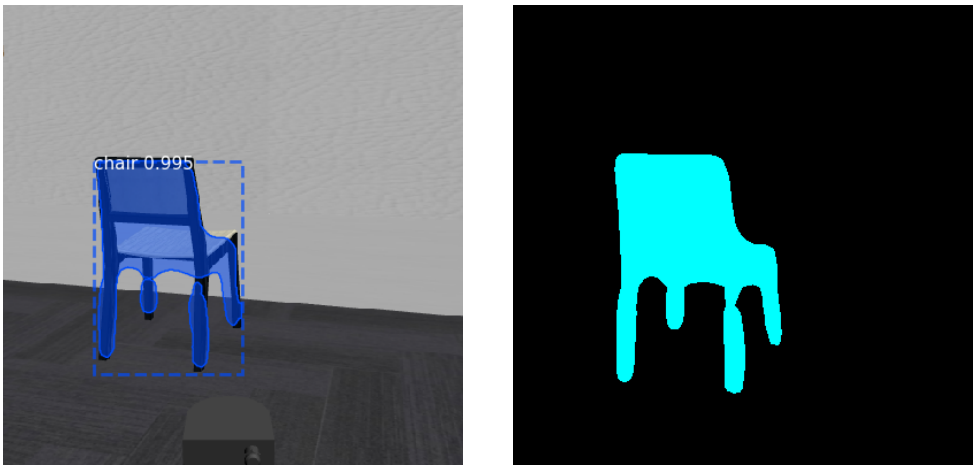
As discussed in section 2.2.1, Mask R-CNN differs from Faster R-CNN only in the second stage. It generates binary masks for all the detected objects in the image parallelly with bounding box detection. Unlike other popular segmentation methods, Mask R-CNN uses per-pixel *sigmoid* instead of per-pixel *softmax* to represent the class inside the mask avoiding a competing effect between classes to occupy the mask. Such representation helps the network to distinguish between multiple objects of the same class.

Implementation Details

The ROS implementation of Mask R-CNN requires only the RGB image provided to it through a ROS topic. As an output, it provides object classification, bounding box, the confidence of prediction, and the mask enveloping an object per image published as a ROS topic with 'bgr8' encoding without the alpha channel. As per the authors, the algorithm can run averaging at 5 fps. Since the depth image and camera_info topics published from the camera node are much faster, they should be synchronized with the segmented image to match with its low-frequency generation. Modifications to the current implementation were made to satisfy the requirements of the semantic reconstruction method used.

To remove the background and to change the encoding to the required 'bgra8' a numpy nd array of size (image_width, image_height, color_channels) was created for every received RGB image input. Each color_channel represents red, blue, green, and alpha. The red, blue, and green channels are initialized with zeros and the alpha channel is populated with 255. The binary masks contained in an image are extracted and iterated through, during which the color composition as mapped in the .csv file of Kimera is multiplied with the mask and added to the red, blue, and green channels. After iterating through all the masks the red, green, and blue channels contain all the objects with the mapped colors and without any background. This nd array is converted into a ROS message with 'bgra8' encoding and is published as a ROS topic.

The node is also modified to subscribe to topics related to depth image and camera_info. The message_filter is a utility library in ROS which has adapted



(a) Original output. (b) Output after modification.

Figure 6: Mask R-CNN output.

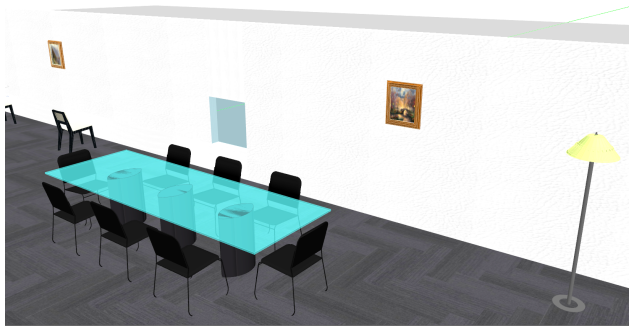
ROS versions of message filtering algorithms. The 'ApproximateTimeSynchronizer' is an algorithm used to filter a set of messages and publish only those which have timestamps within a defined tolerance. The depth image, camera_info, and the modified segmented image are passed through this filter and these approximately synchronized messages are passed in as the input for the Kimera module. Figure 6 compares the output of segmentation before and after the modifications.

3.3.3 Semantic reconstruction

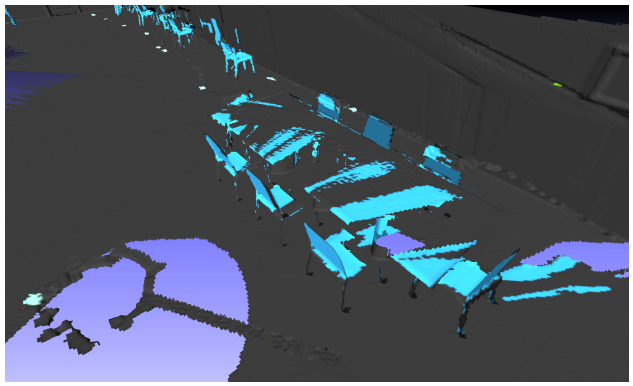
The semantic reconstruction module is responsible to integrate the semantic information of the objects in the environment into the geometric mesh. Semantic information is gathered through a deep learning based pixel-level instance segmentation algorithm, the output of which along with the respective depth image and pose information is passed on to a library that generates the semantically annotated 3d mesh. Kimera and Voxblox++ [28] were considered for this task as both are based on the Voxblox framework which avoids any compatibility issues that may arise between the geometric and semantic-geometric outputs when totally different components are used to generate them. Kimera had the edge over the Voxblox++ in runtime performance with a reported rate of 10Hz whereas the latter reported a rate of 1Hz and hence Kimera was chosen for this task. An already available ROS implementation was also an added advantage.

Since Kimera uses Voxblox for the TSDF construction the working principle remains the same for mesh construction. Additionally, Kimera uses 2D semantic labels to annotate the 3D mesh which can be obtained through any segmentation algorithm. At each keyframe, the generated 2D labels are attached to the 3D point cloud which is used in grouped raycasting. During this process, a probability vector is created based on the frequency of the observed labels in respective groups. At each voxel, the label probabilities are updated through a Bayesian update. At the end of grouped raycasting, the most likely label is extracted from the probability

vectors of each voxel which is used to annotate the TSDF layer. Finally, using the marching cubes algorithm a semantically annotated geometric mesh is constructed.



(a) Simulation.



(b) semantic reconstruction.

Figure 7: Example output of kimera in simulation.

Implementation details

A custom launch file in ROS was created with all the parameter settings the same as that of voxblox except for the RGB image topic. The RGB image topic needs to be remapped to the output of an instance segmentation algorithm. Kimera uses a .csv file to map the objects to their corresponding color representation and as per the user's environment settings this needs to be modified accordingly. Additionally, Kimera requires the input image with 'bgra8' encoding (an OpenCV [58] image encoding) that includes only the labeled objects. Kimera also has the 'generate_mesh' service through which the semantically annotated mesh can be saved in a .ply format. Figure 7 shows an example output generated by the Kimera library in simulation.

3.4 3D Object instance segmentation

Most of the popular 3D data processing libraries favor point cloud format, hence the semantically annotated 3D mesh is converted into a point cloud before subjecting it into further stages down the pipeline. Open3D [59] is a 3D processing library with support for C++ and Python. It can handle both mesh and point cloud type data

and has implementations of many widely used algorithms as packages to extract information, manipulate or transform the 3D structure depending on the purpose. The semantic mesh surface is uniformly sampled using a sampling algorithm in the Open3D library to generate the point sets. Furthermore, all the uncategorized points (without a semantic label) are removed and the resultant point cloud is saved in a .pcd format. Figure 8 shows the output after conversion of mesh to point cloud.

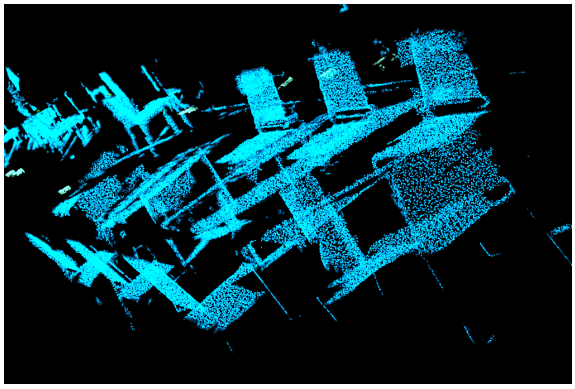


Figure 8: Example output of mesh to point cloud conversion.

The processed point cloud consists of only the recognized objects with color pertaining to their respective class. In order to perform any shape completion strategy, the scene needs to be broken down into individual object instances. Point Cloud Library (PCL) [60] offers a wide variety of segmentation algorithms for the point clouds. The difference of normal algorithm in PCL used for clustering is presented below.

3.4.1 Difference of normals (DoN) method

This method provides a simple and yet computationally efficient method to process large unorganized 3D point clouds. Firstly, unit surface normals for all points are estimated for a given small and large support radius. Secondly, the difference of normals is computed and the resultant vector field is used to isolate the points region-wise based on a threshold. Finally, the Euclidean cluster extraction algorithm is used to extract individual object instances and save them as .pcd files.

Implementation details

This is a C++ executable that takes 4 inputs. A .pcd file, small support radius, large support radius, filter threshold, and distance threshold. However, the algorithm does not retain the color information of the instances. With trial and error, the parameter values were chosen, such that reasonable clusters of individual objects are obtained. Figure 9 shows the cluster of a chair extracted from the scene.

- **small support radius:** 1
- **large support radius:** 2

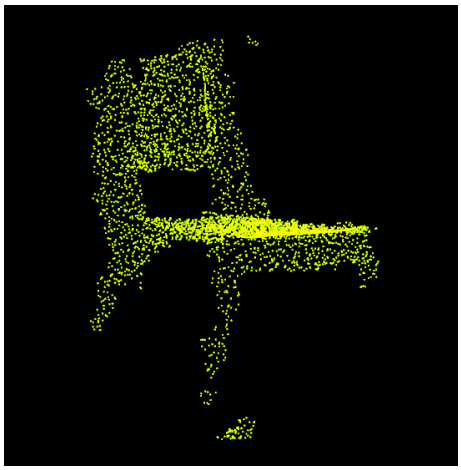


Figure 9: Extracted cluster of a chair.

- **filter threshold:** 0.0001
- **distance threshold:** 0.1

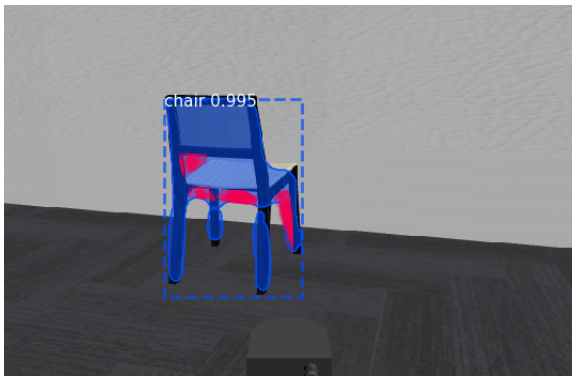
3.5 Scene augmentation module

This is the final and most crucial stage in the pipeline where it is attempted to improve the object representations in the current reconstruction. The purpose of the module is to update the clustered semantic point cloud by replacing object instances with better reconstructions. As discussed in section 2.4 the complete object shape can be inferred by either predicting the missing part of the object or by replacing the object with a matching synthetic model from a database. Different kinds of chairs in the environment were chosen as the candidate object with which the performance of the pipeline would be analyzed.

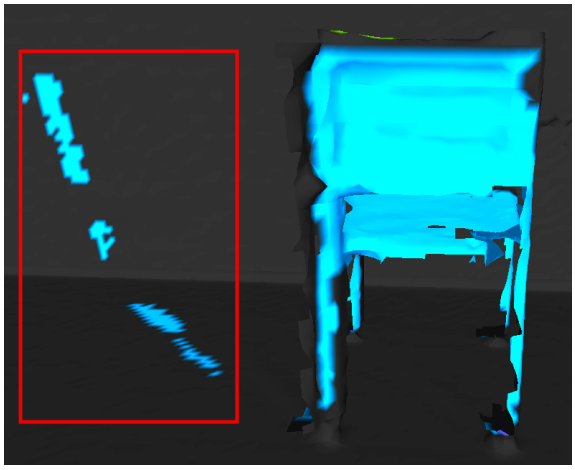
3.5.1 Shape filter

Prior to this stage, the scene point cloud has been broken down into individual object clusters, but some of the clusters may not be relevant. This is because the mask obtained for an object may contain parts that do not belong to the object. Figure 10a shows an example of such a case. The marked regions in the figure do not belong to the object, so when this image and the corresponding depth image are put together to construct a mesh two types of discrepancy might arise. First, when the object is in isolation, part of the walls or floor may get registered as part of the object as seen in Figure 10b. Second, when an object is in close proximity with other objects, parts of those objects or an entire object may be misrepresented as seen in Figure 10c.

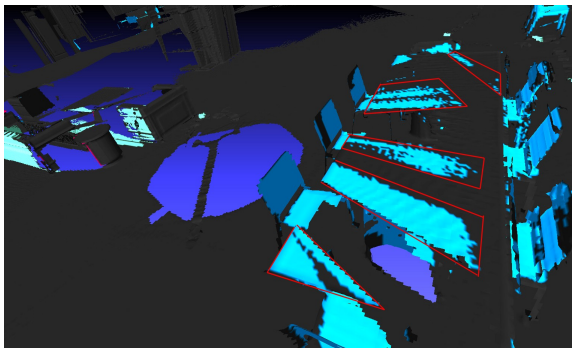
Since it is not possible to alter the performance of Mask R-CNN, avoiding such situations is improbable, thus the solution lies in identifying and filtering out these discrepancies before they are sent for completion. While it is difficult to isolate all the discrepancies, however, it is possible to mitigate them to a certain extent.



(a) Marked regions are not part of the chair.



(b) Marked region denote spread of label onto the wall and floor.



(c) Marked regions denote spread of label onto a table.

Figure 10: Discrepancies caused by inaccurate object mask.

Removing parts of walls and floors

The key factor to notice here is that, since both floors and walls are planar the corresponding point cloud is also planar. Due to the planarity, there will be no points registered along with any one of the X, Y, or Z-axis in the point cloud, such a point cloud can be identified through its covariance matrix. A row and a column

corresponding to the axis where no data is present will be zeros since correlation with itself and other axes are null. If a cluster is encountered with this property, it is ignored from the subsequent stages of the pipeline.

Removing misrepresented objects

Identifying misrepresented objects is difficult as the misrepresented object most often does not contain any one particular unique trait. However, the distance of the farthest point from the centroid of any sort of chair (point cloud representation) commonly found in an office or home environment would be lying in a certain range. So any misrepresented objects falling out of this range can be removed. Nevertheless, this only removes the wrong object if its size is much larger than that of a chair. Smaller or comparable objects still remain in the subsequent stages and result in becoming false positives.

Since the next stage in the pipeline only expects partial point clouds of chairs, object clusters with labels other than that of the chair are also omitted.

3.5.2 Shape completion using a learning-based method

The function of the learning-based method is to identify the gaps in the geometry of the input objects and complete the missing regions. Since the object clusters are in point cloud format the method had to be compatible with point cloud data. Additional requirements were that the method should be able to handle the object classes that the Mask R-CNN could recognize and the output should have high resolution. Since PCN [46] served as the benchmark for all the newly developed methods that work in the point cloud domain, PCN was chosen for the intended task. Moreover, the following attributes of PCN increased its viability to work alongside the pipeline.

- State of the art method for point cloud completion task
- Implementation in Python is available
- Network weights trained on 8 ShapeNet model class: airplane, car, chair, lamp, sofa, table, vessel is available
- Output consists of 16384 points
- Can directly work on the object clusters obtained from the segmentation stage
- Pose information from the partial input is retained in the output

PCN is a supervised learning approach that leverages ShapeNet models to create a dataset consisting of pairs of partial and complete point clouds for training the network. Figure 11 shows the network architecture used by PCN. It is an encoder-decoder network that takes a partial point cloud as input and produces a complete point cloud as output. The encoder consists of two stacked PointNet [45] layers. The first layer constructs a feature vector from the partial point cloud. A *maxpooling*

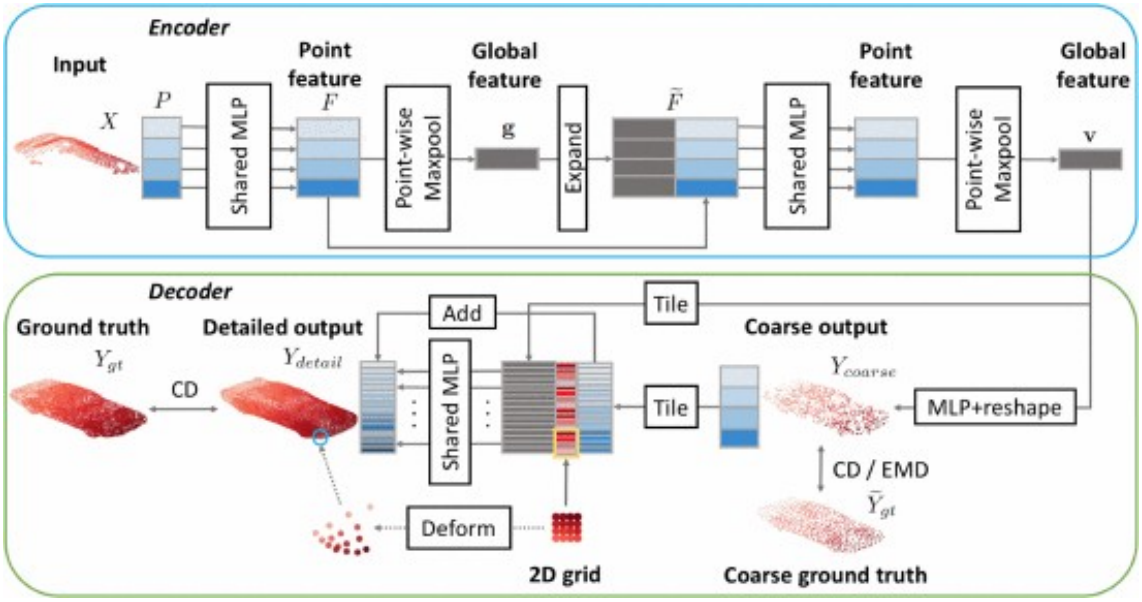


Figure 11: PCN network architecture. [46]

operation on individual points is performed to create a global feature vector. Both these feature vectors concatenated together form the input for the second PointNet layer. A *maxpooling* operation on the output of this layer gives the final feature vector. This layered approach helps to capture the detailed geometric information in the input point cloud. As for the decoder, PCN has a multistage decoder which is a combination of a fully-connected decoder that is capable of predicting the global geometry and a folding-based decoder [61] which performs well in generating smooth surfaces capturing the local geometry. The final feature vector from the encoder acts as the input to the fully-connected decoder which generates a sparse point cloud that represents the global geometry. This sparse point cloud along with the final feature vector is passed through the folding-based decoder to generate a dense point cloud as the final output.

Implementation details

The PCN network expects the input point cloud with its centroid located at the origin and the points arranged in (YZX) order. Since the segmented point cloud clusters from the DoN method retained the pose information in the world, a python script was written to accommodate the requirements put forth by the PCN along with other helper functions. As mentioned in the section 3.4.1 object clusters did not retain the color information. Hence, a random point from each cluster is sampled and is compared against the scene point cloud which also has the same exact point to retrieve the label of that object cluster. The input clusters would then be subjected to the shape filter implemented as explained in section 3.5.1. Each of the successful clusters is translated to the origin after their actual location in the world has been stored in memory. They are reordered to have (YZX) structure before passing through the network to obtain a complete point cloud. Points in each output are

reordered to the original (XYZ) structure and translated back to their location in the world. The Open3D library was used to process the point clouds. Figure 12 shows an example of output by PCN.

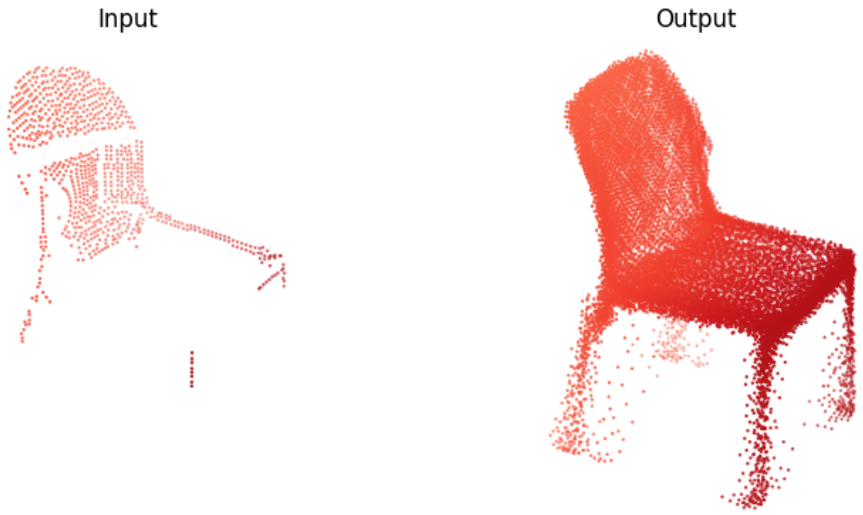


Figure 12: Example output of object completion by PCN.

3.5.3 Model matching method

In this thesis, as an alternative to learning-based shape completion, it was attempted to identify a similar synthetic model of objects which could take its place in the reconstructed mesh to reduce the ambiguities involved regarding the unobserved parts of the objects. The steps involved to achieve this are as follows:

- Create a database of synthetic models of interest
- Match the pose of the model to the partial object
- Match the partial object against the database to find a similar model

Implementation details

All the steps are achieved exclusively using Python. Kaolin [62] and Open3D libraries were used to process the mesh and point cloud data.

Creating a database

As mentioned in Section 3.5 types of chairs were chosen to be the objects of interest. ShapeNet was the source of our database since it possesses a collection of 6778 3D models of chairs. Kaolin [62] is a PyTorch library for the purpose of accelerating 3D deep learning research. It serves as a tool for processing and loading 3D data in an accelerated manner. It can handle a wide variety of popular 3D data representations

such as voxels, meshes, point clouds, and their file formats. Its functionalities make it easy to convert 3D data in any format into any other format as required by the user. Because of these reasons Kaolin was used to create the database.

ShapeNet models come with the .obj file format. Kaolin has packages that can read .obj files and process them. For conversions to point cloud it is possible to set the required resolution. A resolution of 10000 points was set for each of the chair models. The point clouds are saved in a directory in a .pcd file format same as how the partial object point clouds were saved.

Matching the pose of the model to partial object

Inferring the pose of an object directly from its raw point cloud representation is quite difficult. Instead, it is easier to find a transformation that can be applied to the model that will align it with the object through point cloud registration methods. Iterative Closest Point (ICP) is a well-known algorithm that, given source and target point cloud, performs transformations (rotations and translations) on source point cloud iteratively, reducing the distance from the source to target and attaining the best possible registration between the two point clouds.

The input partial point clouds have their points in the world frame. Hence, the centroid of the point cloud is subtracted from each of the points. As a result, the object representation will be in its local frame centered at the origin. The same procedure is followed for all the database models as well before subjecting both to the registration algorithm. Also, both sets of point clouds are normalized to be within the range (0,1) to match the scale.

Figure 13: RANSAC solution for coarse alignment when one leg was missing.

ICP is a computationally expensive algorithm and is susceptible to getting stuck at local minima resulting in suboptimal alignment solution, so it is best if any initial estimate of the transformation is available to loosely align the source (database models) with the target (partial point clouds). The ICP can then fine-tune the

coarse transformation in very few iterations to tighten the alignment. A way to get a coarse alignment between point clouds is by using global registration methods. RANdom SAmple Consensus (RANSAC) based methods for point cloud registration falls under this category. A RANSAC based method in the Open3D library was tried in the thesis to obtain an initial estimate of the transformation. First, the Fast Point Feature Histograms [63] (local shape descriptor) for the source and target are computed. Then a random set of points from the source is sampled and their corresponding points in the target are determined using nearest neighbor search. A transformation between the source and target is computed and validated on the entire point cloud with a distance threshold. This set of steps are repeated for a predefined number of iterations to get a transformation that coarsely aligns the source with the target. However, since the method heavily relies on local geometry, the quality of the solution depends on the completeness of the object point cloud. Hence, sometimes it fails to find the correct solution when the object point cloud has some important parts missing in it, Figure 13 shows an example of a situation where the algorithm fails to find the correct solution. Since the input is missing a part of the leg the local geometry captured mostly has attributes of the seat and the backrest which influences the solution to miscalculate and align the model around the known geometry.

Alternatively, a far more simple and yet effective approach was found. The key idea was that since the chairs are grounded objects, their natural pose will always be around the Z-axis. A random model from the dataset is selected and a set of uniformly sampled rotations are performed around its Z-axis. The average point to point distance between the model and object point cloud is computed for each rotation and saved in an array. After the end of rotations, the transformation matrix for the rotation that yielded the minimum average distance is calculated which serves as the estimate for the ICP algorithm. ICP fine-tunes this transformation to provide the final transformation matrix. However, blindly using the ICP refined transformation can result in situations where some of the legs of the model are not touching the ground. This happens in situations where the partial object lacks one or more legs and the ICP tries to finitely align the synthetic model with each part of the partial object. To mitigate such situations the transformation matrix obtained after the ICP is decomposed into rotations around X, Y, and Z axes. The rotations around the X and Y axes are ignored and the final transformation matrix is recalculated with just the rotation around the Z-axis such that all the legs of the model would be grounded in all situations. The advantage of this approach is that it is applicable to any of the grounded objects. Figure 14 shows an example of the final alignment obtained with this approach. Blue represents the model and yellow represents the input object.

Finding a match

Finding a match involves searching through the database. Since all the database models are represented with 10000 points, searching through the database can become time consuming. Hence, the models are uniformly downsampled by a factor of 20

(a) Front view. (b) Side view.

Figure 14: Alignment with simplified approach.

(a) Input. (b) Output.

Figure 15: Example output of model matching method.

preserving the local geometry thereby reducing the computational burden. The final transformation refined after the ICP method is applied to each model and the corresponding point to point distance with the partial point cloud is recorded. The hypothesis is that the model that most closely resembles the actual object should return the least distance. However, this does not guarantee a well-matched object always since it depends on the amount of completion in the object point cloud. Figure 15 shows a visual comparison of the actual object and the matched model. The matched model in the local frame of the partial object is rescaled and translated back into the world frame and stored in an array for further stages.

3.5.4 Object replacement

This is the final stage of the pipeline where we remove the objects that are currently in the environment and replace them with either a completed version from PCN or with a similar synthetic model. This task involves two subtasks, removing the objects and replacing the objects.

Implementation details

The implementation is completely using Python extensively using Open3D libraries. The functions are written to perform two tasks: remove the partial objects from the scene, replace the objects with either the outputs of PCN or the models that were matched with the objects back into the scene.

Removing objects from the scene

The scene referred to here is the one created with the Voxblox library. First, a numpy array of the point cloud representation of the scene is created. KD-trees are built around the centroid of all the objects of interest. The index of points spanning a search radius around these objects is fetched and deleted from the numpy array of the scene point clouds. However, this approach is not ideal as it removes some background information from the scene.

Replacing objects into the scene

The output from the previous stage (from PCN or model matching) is a numpy nd array holding the complete representation of deleted objects. This nd array is appended with the numpy array of the scene without objects of interest to create the augmented scene with a complete representation of said objects.

4 Experiments and analysis

The experiments devised to evaluate the pipeline answer the following questions:

- How does the performance of the PCN method compare to the Model matching method?
- How is the performance of the pipeline as a whole?
- Which is the weakest module in the pipeline that affects the performance of the pipeline as a whole?
- How dependent the performance of the pipeline is on the speed of the robot during mapping processes?
- How well does the pipeline perform in the real world with noisy sensor data?

4.1 Simulation

Figure 16: Top view of the office environment in Gazebo.

An office environment [64] setup in Gazebo [65], a robot simulation toolbox, was used to test the pipeline. The office environment consists of a total of 19 chairs. 8

chairs around a long meeting table, 6 chairs facing the wall with some objects in between them, 4 office chairs facing a table and partially slid inside, and one chair near a long table facing the room. Figure 16 shows the top view of the office setup. As for the robot that would navigate the said environment, a Husky robot model from 'CLEARPATH ROBOTICS' equipped with a Kinect RGBD camera and a SICK laser scanner was set up. Figure 17 shows the robot model with sensor mounting. A PC with Intel Core i7-8750H, 2.20GHz CPU, Nvidia GeForce 1050 Ti GPU, and 16 GB RAM was used to conduct the experiments.

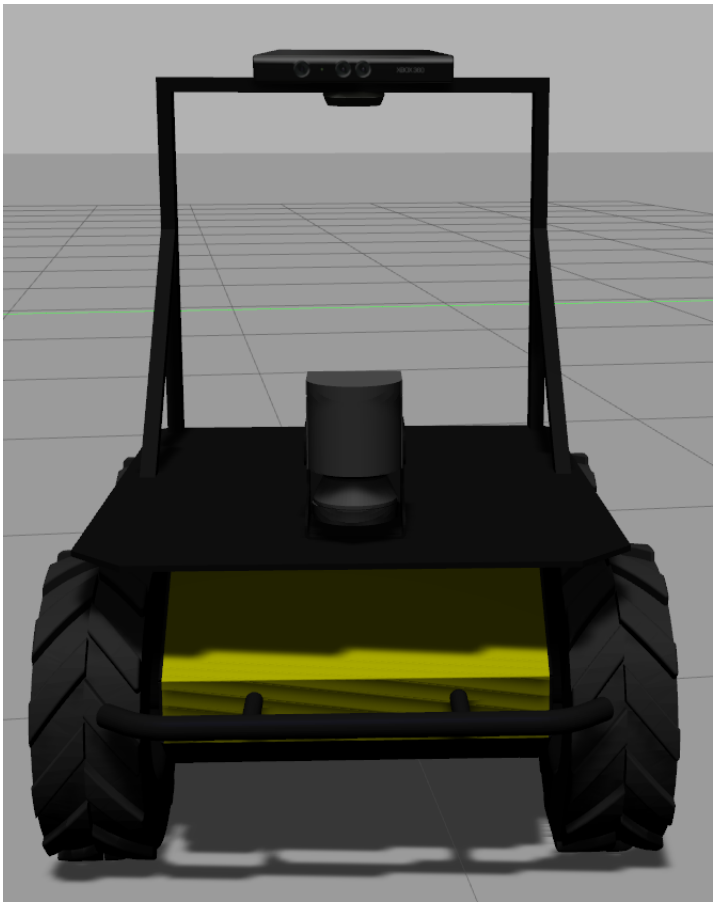


Figure 17: Husky robot model.

4.1.1 Model matching vs PCN

Model matching, a novel approach developed in this thesis, and PCN, a learning based method were the two methods that were explored to replace the partially known objects with their respective completed versions. To determine which method provides better results two tests were devised for the evaluation: Based on the execution time of each method and the quality of reconstruction.

The robot was navigated through the environment with 19 chairs and a set of geometric and semantic mesh was created. This served as the input for both the methods.

Execution time

For both the methods, time taken from the point of receiving the object clusters from the scene segmentation till the final scene is recreated was measured for analysis. Table 2 shows time taken to augment the scene and deliver the final output for each method.

Method	CPU time (h:m:s)	Wall time (h:m:s)
PCN	00:01:36	00:00:47
Model Matching	00:27:18	00:03:07

Table 2: Execution time for PCN and Model matching methods.

Analysis

Based on Table 2, it is quite clear that the PCN method is much faster than the model matching method. The Model matching method has quite a few bottlenecks:

- Loading the database into memory
- Searching through the entire database to find a match for each input object
- Pose estimation

Among these 3, loading the entire database into the memory is the most time consuming part because there are 6778 model files to read each with 10000 points. Nevertheless, this can be handled through efficient parallelization of CPU cores using multi-threading libraries in python which can accelerate this process. Evidently, the CPU usage time is very high compared to that of wall clock time since the database handling is shared between all the CPU cores ensuring complete utilization of the CPU capabilities. The process of database search to determine the match cannot be parallelized as maintaining the sequence of the database chairs is important to correctly identify the match. Hence the time taken to complete this process depends on the number of chairs under consideration and the vastness of the database. PCN does not have any of the above bottlenecks as all the above functions are carried out by a trained neural network without the aid of any database.

Based on quality

In the context of the pipeline, the quality refers to how well the completed output relates to the input. There are 19 chairs in the environment whose ground truths are compared against the outputs of PCN and Model matching. False positives are not considered in this analysis. To systematically analyze the results, the following set

of questions were devised based on which outputs of both methods will be compared to the corresponding Ground Truth (GT):

1. Does the support comprise a legged structure?
2. Does the support comprise a central axis with a base?
3. Does it have arms?
4. Does it have a backrest?
5. Does the backrest have any hollow area?

Table 3 provides comparative results of the PCN method and the Model matching method against the ground truth. The 5 questions listed above characterizes the ground truth chairs which are compared to the outputs of PCN and Model matching. The blue color indicates the data is too noisy or lacks any structure for that particular aspect. The green color indicates a match with the ground truth and the red indicates the presence or absence of a characteristic in the ground truth. For e.g, consider row 2, the ground truth chair has 4 legs, no arms, and has a backrest without hollow area. The PCN output corresponding to this chair lacks the structure that defines legs, has a backrest without hollow area, and no arms. For model matching output, it has 4 legs, a backrest without hollow area, and has arms different from that of ground truth hence marked in red.

Chair	GT					PCN					Model matching				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	✓	✗	✗	✓	✗	●	●	✓	✓	✗	✓	✗	✗	✓	✗
2	✓	✗	✗	✓	✗	●	●	✗	✓	✗	✓	✗	✓	✓	✗
3	✓	✗	✗	✓	✗	-	-	-	-	-	✓	✗	✓	✓	✗
4	✓	✗	✗	✓	✗	●	●	●	●	●	✓	✗	✓	✓	✗
5	✓	✗	✗	✓	✗	●	●	●	●	●	✓	✗	✗	✓	✗
6	✓	✗	✗	✓	✗	●	●	●	●	●	✓	✗	✗	✓	✗
7	✓	✗	✗	✓	✗	●	●	●	●	●	✓	✗	✗	✓	✗
8	✓	✗	✗	✓	✗	●	●	●	●	●	✓	✗	✓	✓	✗
9	✓	✗	✗	✓	✓	✓	✗	✗	✓	✗	✓	✗	✗	✓	✗
10	✓	✗	✗	✓	✓	✓	✗	✗	✓	✗	✓	✗	✓	✓	✗
11	✓	✗	✗	✓	✓	✓	✗	✗	✓	✗	✓	✗	✗	✓	✗
12	✓	✗	✗	✓	✓	-	-	-	-	-	-	-	-	-	-
13	✓	✗	✗	✓	✓	✓	✗	●	●	●	✓	✗	✗	✓	✓
14	✓	✗	✗	✓	✓	✓	✗	●	●	●	✓	✗	✗	✓	✗
15	✓	✗	✗	✓	✓	-	-	-	-	-	-	-	-	-	-
16	✗	✓	✓	✓	✗	-	-	-	-	-	-	-	-	-	-
17	✗	✓	✓	✓	✗	-	-	-	-	-	-	-	-	-	-
18	✗	✓	✓	✓	✗	●	●	●	●	●	✗	✓	✓	✓	✗
19	✗	✓	✓	✓	✗	-	-	-	-	-	-	-	-	-	-

Table 3: Visual quality results of object replacement from PCN and Model matching.

Analysis

Referring to Table 3 the output of the PCN network is very inconsistent, mostly noisy, and unusable. The hypothesis for this behavior was that the network may have overfitted with the training data. To further investigate this, the output of the network for the training data was compared with the output of the partial data of objects from the simulation environment.

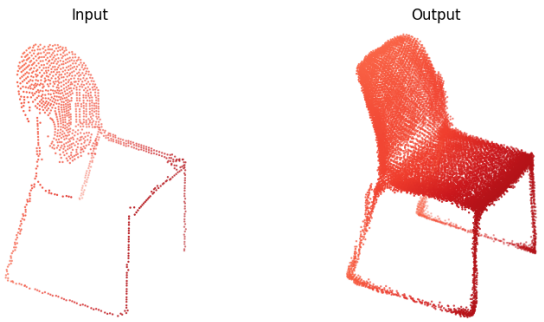
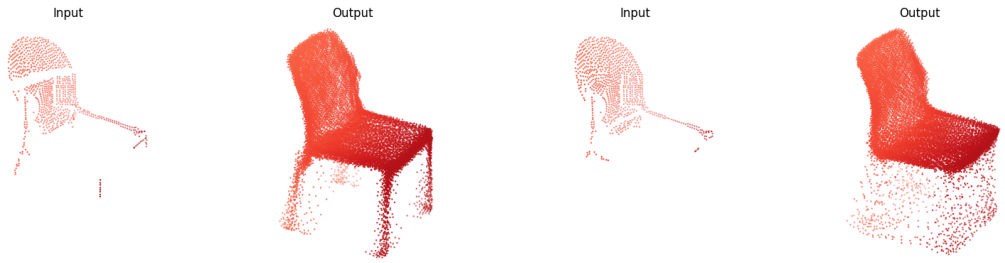


Figure 18: Completion of a chair from the training data.

As mentioned in Section 3.5.2 the network was trained on ShapeNet models. Custom datasets of pairs of partial and complete point clouds extracted from ShapeNet models were used for training. For the complete point clouds, 16384 points were uniformly sampled from the mesh surfaces of the models. The Partial point clouds are generated by back-projecting 2.5D depth images into 3D. For each model 8 partial point cloud was generated from 8 random viewpoints and paired each one with the complete point cloud of the model. In effect, the network uses the different random orientations of an object to map the partial view to the complete object and learn to complete the missing region from that view. This type of training might be the reason for the inconsistent behavior of the network as it might be susceptible to the percentage of missing information in the object or the pose of the object which the network might not have seen during training. Figure 18 shows the completion of a chair from the training data. The completed output is fairly similar to that of the partial input without any significant outliers. However, when the part of the legs was removed, the network is less confident about the structure and the completed region has noticeable outliers as seen in Figure 19a. Furthermore, when the input has no legs the network falls apart and the completed region is full of noise as seen in Figure 19b.

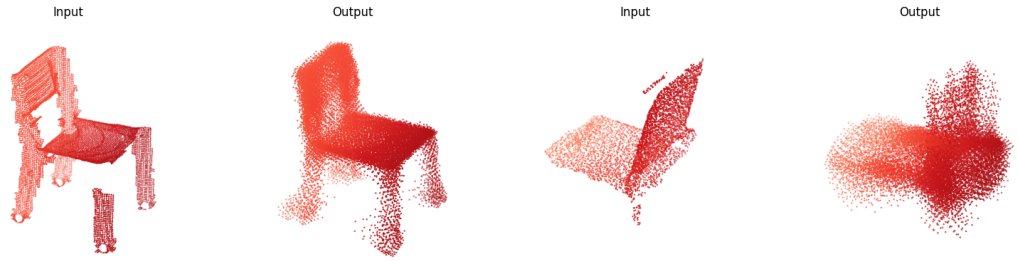
Moving to the data from the simulation, referring to Figure 20a the input object is lacking very few regions but still, the output produced by the network has visible outliers. When a similar chair in a different pose as in Figure 20c is subjected to the network, the confusion of the network is quite evidently seen as it tries to recreate the backrest resulting in a box-like structure above the legs. Figure 20b and 20d are the same chairs in different orientations. For the former, the network is unable to map it to a chair whereas for the latter it is not able to infer the extent of legs with one partial leg. All these observations point to the fact that the network may have indeed



(a) Part of the legs removed from the input.(b) Legs completely removed from the input.

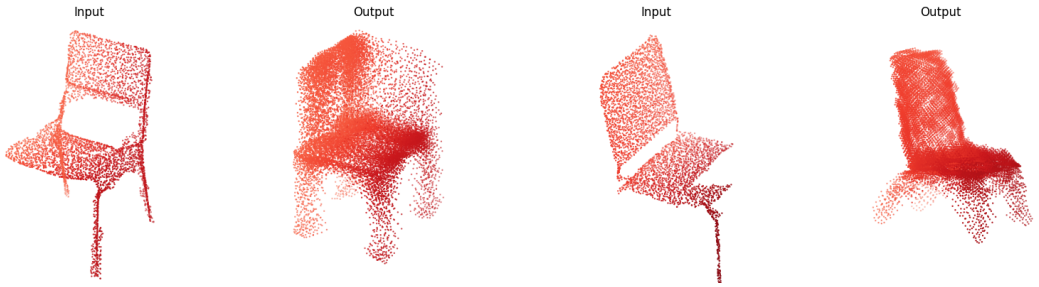
Figure 19: Failure cases of PCN.

overfitted with the training data and is not able to produce satisfactory results on external data. Table 3 shows that majority of the chairs lack proper structure, some in parts like chair 13, 14 have distinguishable legs with the upper structure being filled with noise, such an example can be seen in Figure 20c and some others with completely unstructured geometry similar to Figure 20b. Chairs like 9, 10, 11 have reasonable reconstruction as the input had very few missing regions like in Figure 20a. However, there are outliers present even in favorable reconstructions.



(a) Chair with very low missing region.

(b) Chair with no legs.



(c) Chair with 3 legs.

(d) chair with 1 leg.

Figure 20: Performance of PCN on data from simulation.

The inconsistencies such as noise and outliers will not be present with a model matching system. In this case, what influences the output quality is the region of missing information rather than its quantity. When the input has parts of all the factors that define a chair which are legs, backrest, seat, and arms a good approximate match is obtained as seen in Figure 21. However, when there is a key component

(a) Input. (b) Output.

(c) Input. (d) Output.

Figure 21: Matched models when the input has data for backrest, seat, and leg/legs.

(a) Input. (b) Output.

(c) Input. (d) Output.

Figure 22: Matched models when the input has a key component missing.

completely missing from the input the match found would not be quite so similar to

that of the actual object. Consider Figure 22a which is an office chair with wheels. However, the input lacks the arms and the wheeled base is not captured well which reflects on the matching model in Figure 22b which does not have arms and has a circular base instead of a wheeled base. For the input in Figure 22c where only a partial extent of one leg is present the corresponding match found has very short legs and when this model is placed back in the scene would hover over the ground. The output also has a set of arms that are not present in the actual object. From the observations, it is clear that a complete extent of at least one leg is desirable in the input to find a good match that would be properly grounded.

Referring to Table 3, the first 8 chairs surround a table close to each other. Models matched for chairs 2, 3, 4, and 8 had short legs as the input failed to capture the full extent of at least one leg. The rest of the model matches are quite similar to the actual object.

Conclusion

The PCN method outperforms the model matching method when it comes to execution time. However, it seriously falls behind the latter when it comes to the quality of reconstruction. Referring to Figure 20, PCN showed to be highly susceptible to pose variation and amount of missing information and even the ones with proper completion had significant outliers in the structure making it impractical to employ this method outside the training data. Since the model matching is completely deterministic the output it produces will always be noiseless with definitive structure as opposed to the learning based method, but still, it is not without flaws. Since the method always tries to find the closest match, if any key component is completely missing from the input the method would have no knowledge of its existence and will be reflected in the match found. Nevertheless, as long as all the key factors that define the particular object are present to a certain degree the method will find a reasonable match for that object.

Since the analysis shows that PCN output to be of inferior quality the rest of the analysis is carried out only with the Model matching method.

4.1.2 F_1-score analysis

F_1-score is a weighted harmonic mean of Recall (R) and Precision (P), where Recall measures the ratio of the number of correctly identified positive samples to the number of all positive samples and Precision measures the ratio of the number of correctly identified positive samples to the number of all identified positive samples which will include all the false positives as well. The highest possible value of a F_1-score is 1, indicating a perfect Precision and Recall, and the lowest possible value is 0 when either the Precision or the Recall is zero. This is a commonly used metric in the fields of Information Retrieval and Machine Learning, especially when the measurement concerns only one class like in this case where only chairs are considered as objects of interest. In Section 4.1.1, the quality of model retrieval was analyzed, however, that only analyses the performance of the model matching method alone. F_1-score is a way to capture the performance of the entire pipeline as a whole and

possibly identify the weak links in the pipeline. However, this method does not consider the effects of false negatives. Nevertheless, the case where the pipeline produces false negatives is only when an actual chair is detected to be a different object by the deep learning algorithm. The occurrence of false negatives are quite slim as the state of the art algorithms has great accuracy in object detection.

$$F_1 score = \frac{2PR}{P+R} \tag{1}$$

where,

$$P = \frac{\text{number of correctly identified chairs}}{\text{number of identified chairs}}$$

$$R = \frac{\text{number of correctly identified chairs}}{\text{number of all the chairs}}$$

Results

The final reconstruction of the simulated office environment after the replacement with the models was considered for the evaluation. A total of 16 chairs were identified and replaced. Out of the 19 GT chairs 11 of them were correctly identified and the other 5 chairs were false positives. A Precision of 0.6875 and Recall of 0.5789 was calculated providing an overall F_1-score of 0.6285.

Analysis

The main factors that negatively affect the F_1-score is when actual chairs are not recognized and when there are false positives. These situations are analyzed in detail.

Referring to Section 3.5, a problem of labels spreading to the surrounding space or objects was recognized due to the inaccurate object mask obtained from the Mask R-CNN. Measures were taken to mitigate its effect and a filter was implemented as detailed in Section 3.5.1, but the filter cannot handle all discrepancies. The discrepancies that get past the filter end up being false positives. Figure 24 zooms over the meeting table. It can be seen that portions of the table are missing. It is because of the removal method of actual objects from the scene, where points around the actual object within a radius were removed. Figure 23 provides a top view of the scene after object replacement and the marked objects denote the false positives. The reasons for these 5 false positives are discussed below.

Figure 23: Top view of the final scene, each label denotes a false positive.

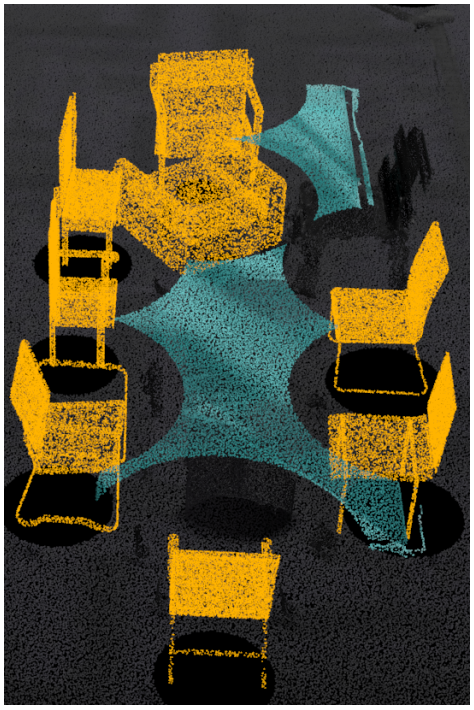
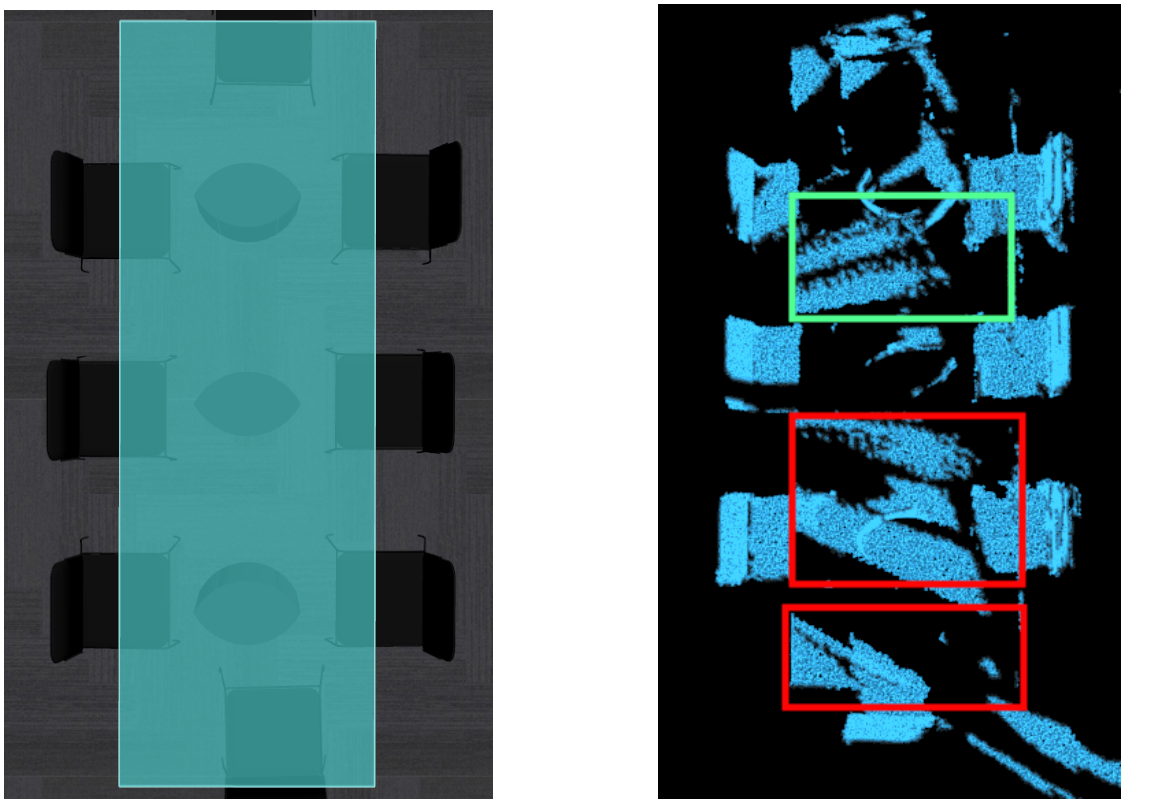


Figure 24: Zoomed in meeting table.

Cases 1, 2, and 5: Label spreading

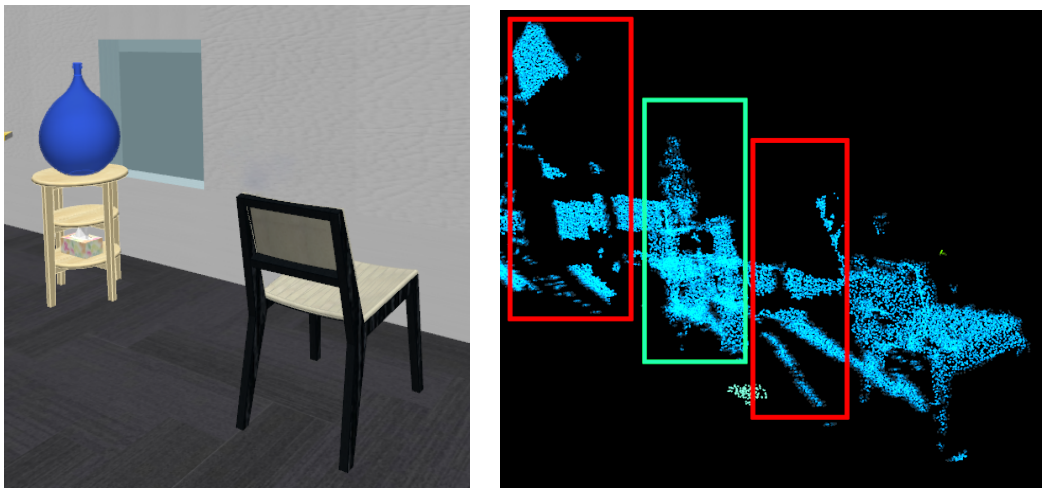


(a) Meeting table in simulation.

(b) Meeting table semantic reconstruction.

Figure 25: Case 1: Meeting table part as false positive.

The shape filter was designed such that it ignores any cluster whose maximum distance of a point from the centroid does not fall in a certain range. However, for cases 1 and 2 the structure of the cluster had a comparable distance from the centroid as that of typical chairs and managed to get past the filter. In Case 1, where part of a table became false positive was the result of label spreading from multiple chairs surrounding the table, and as for case 2 a flower-pot stand got matched with models due to the label spreading from the adjacent chair which fell in the line of sight of the robot while capturing the chair. The detailed view of the input clusters before segmentation is shown in Figure 25 and Figure 26. The regions marked in red were filtered out and green marked regions managed to pass the criterion of the filter to become false positives.



(a) Flower-pot stand area in simulation.

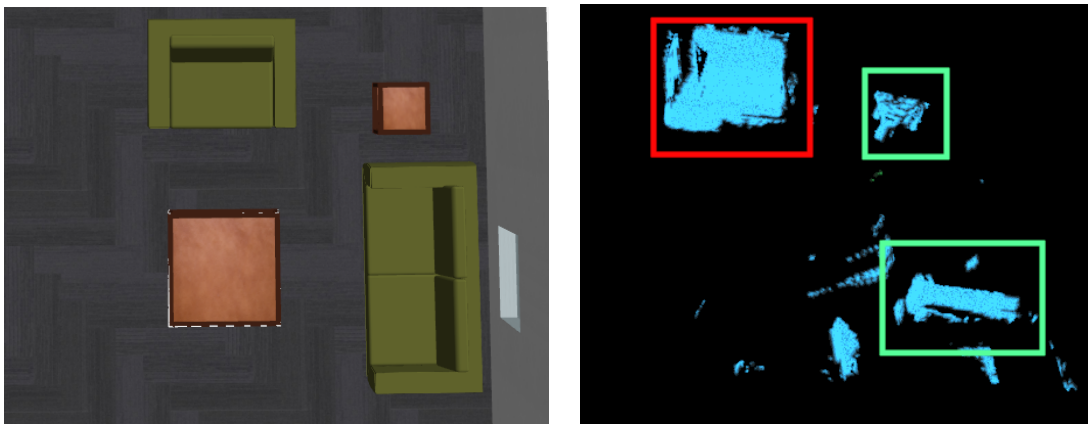
(b) semantic reconstruction of flower-pot stand area.

Figure 26: Case 2: Flower-pot stand as false positive.

Case 5 is very much similar to case1 where a part of the table became false positive due to the label spreading from an adjacent office chair.

Case 3 and 4: False detection

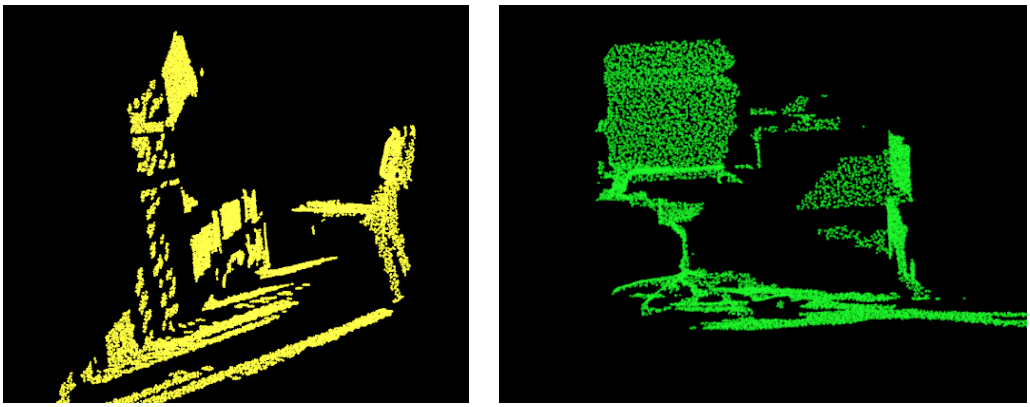
Apart from label spreading, another case where false positives can emerge is when a non-chair object is detected as a chair. This particular scenario is more probable than false negatives as there are objects that can resemble a type of chair such as parts of the sofa, small table. For case 3, a part of the sofa was detected to be a chair while traversing, and for case 4, the side stand was mistaken to be a chair as seen in Figure 27.



(a) Seating area in simulation.

(b) semantic reconstruction.

Figure 27: Case 3 & 4: sofa and side stand as false positives.



(a) Background and the chair clustered together.

(b) Office chair and table clustered together.

Figure 28: Failure cases where actual objects are left out.

The other factor that affects the F_1-score is when a chair object goes unrecognized. Consider the examples in Figure 28, where actual chairs got ignored from the end stages. This can also be attributed to label spreading. In these cases, the cluster segmentation algorithm clustered the object along with the area where the label was spread. Such clusters get filtered out as their maximum distance of a point from the centroid exceeds the set range.

Conclusion

The factors that negatively influenced the F_1-score were analyzed in detail. It is quite clear that the inaccuracy in mask generation around the object is the main root cause that contributes towards either generation of false positives or elimination of objects during the filtering which in turn affects the F_1-score. Hence, Mask R-CNN can be considered as the weakest link in the whole pipeline.

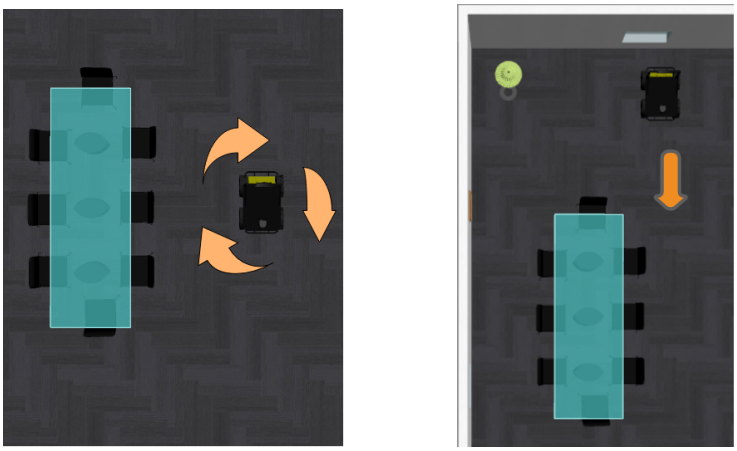
4.1.3 Effect of speed on performance

From Section 4.1.2 Mask R-CNN was identified as a weak link in the pipeline. Apart from the object mask inaccuracy it also suffers from computational overhead as it is a two-stage network. This can be a bottleneck that will affect the quality of the pipeline. Hence dependency of the pipeline performance on the robot speed during data acquisition was analyzed.

Test setup

The F_1-score against different angular and linear velocities of the robot were measured separately. The robot was spawned near the meeting table as seen in Figure 29a and was rotated for one complete cycle from the same initial heading for angular velocities ranging from 0.1 to 0.5 rad/s. Data gathered during each cycle was used to run the pipeline and calculate the corresponding F_1-score. As for the linear motion, the robot

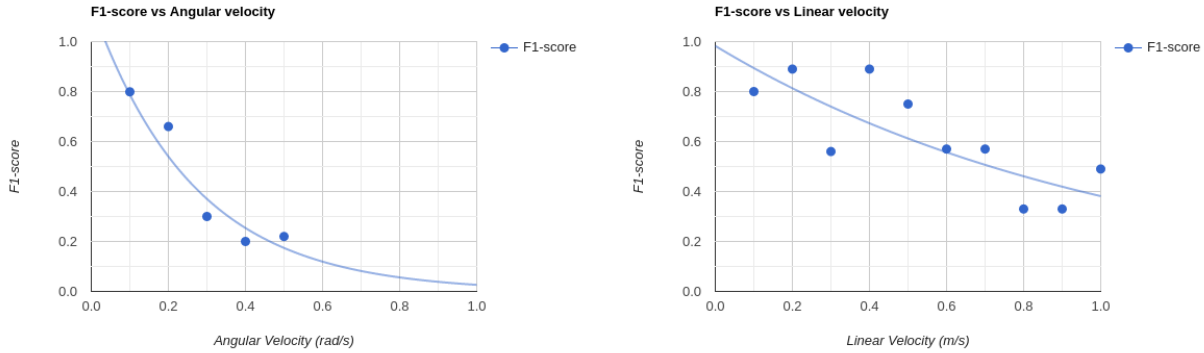
was spawned facing the meeting table as seen in Figure 29b and was moved from the spawned location till the end of the meeting table for linear velocities ranging from 0.1 to 1 m/s. Finally, F_1-score was calculated corresponding to each speed.



(a) Angular motion. (b) Linear motion.

Figure 29: Test setup.

Results



(a) F_1-score vs Angular velocity. (b) F_1-score vs Linear velocity.

Figure 30: Test results.

The results as shown in Figure 30 indicate a degrading performance with increased velocities for both angular and linear motion. The increase in angular velocity has a higher impact on performance, hitting a very low F_1-score for comparatively low angular velocities. The best performance was reported at the lowest possible angular velocity of 0.1 rad/s. The degradation of performance for translational motion is much more gradual than the angular motion registering the highest performance at 0.2 m/s.

Analysis

The performance of the pipeline depends on the amount of data gathered which in turn depends on the sensor input frequency to the reconstruction modules (Voxblox and Kimera). Voxblox takes the raw sensor input directly whereas the Kimera module requires processed input through Mask R-CNN where the bottleneck lies.

The raw sensor data provided by the Gazebo simulator, i.e., camera and pose data, is available at 20 Hz whereas the frequency of the topics published by Mask R-CNN which is the input to the kimera module was observed to be at a very low frequency of 1.1 Hz due to its computational overhead. This means that it takes around 1 sec to segment 1 image and publish it. So at higher angular velocity when the robot view area changes rapidly the kimera module does not receive adequate data to construct a coherent semantic mesh which can be attributed to the rapid decay in performance for faster angular motion. For translational motion, the perceived area of the robot remains more consistent than the angular motion and hence, the gradual decay in the performance.

Conclusion

For the pipeline to give an acceptable level of performance, the robot should move at a meager pace due to the bottleneck created by the image segmentation module.

4.2 Care-O-bot

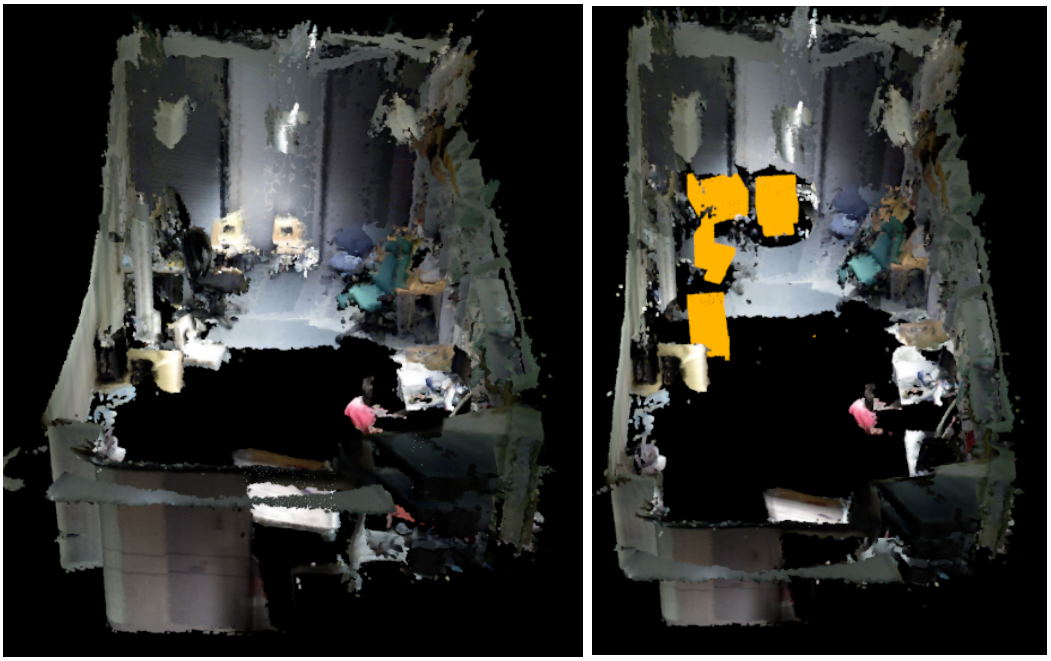
Figure 31: Care-O-bot 4.

The tests on simulation provided an insight into the performance, the factors affecting it, and the bottlenecks in the pipeline. With these limitations in mind,

the viability of the pipeline was tested on a Care-O-bot 4 [66] robot produced by Fraunhofer IPA. The robot was designed to be a mobile assistant to support humans with 29 DoF. It is equipped with multiple RGBD cameras on the head, neck, and torso to gain visibility of the entire area in front of it. It is capable of omnidirectional motion with a maximum velocity of 1.1 m/s. The three laser scanners mounted in the base scan the environment to help the robot react to obstacles, both static and dynamic. For the experiments only the RGBD camera mounted in the neck area was used. Figure 31 shows the front view of the robot. The pipeline is only evaluated with the Model matching method as the PCN method failed to perform well even in the simulated environment.

4.2.1 Care-O-bot in room

7 chairs were laid out in a chaotic manner in the room. The robot was navigated through the environment and the data captured through the RGBD camera mounted in the neck was used to run the pipeline. Figure 32 shows the environment reconstruction and the output of the pipeline. It was pre-configured to only run at a preset low or high speed. The robot was run with low speed for this test setting.



(a) Room reconstruction. (b) Room with chairs replaced.

Figure 32: Before and after chairs are replaced.

There are 7 chairs in the environment and 4 of them were correctly identified and replaced providing a F_1-score of 0.72.

Analysis

From the tests and analysis of simulations following aspects were found to affect the performance of the pipeline:

- Frequency at which the camera images are published
- Frequency at which the segmented images are published
- Speed of navigation of robot within the environment
- Object mask accuracy in the segmented images

The object mask accuracy and the frequency of segmented image output solely depend on Mask R-CNN. As for the camera images from the robot, the RGB images were getting published at 10 Hz and the depth images at a lower frequency of 7 Hz. The robot was operated via a joystick controller which is preset to run the robot at a low or high speed. The Mask R-CNN publishes the segmented image at 1.1hz.

As determined in the simulated environment the best possible results are obtained when the pace of the robot is minimal especially during angular motion. Even though the data was captured at a low speed, it is much higher than the desired speeds. So when the low-frequency camera-input gets coupled with moderate velocity imparts negative effects on reconstructed scenes. Consider Figure 33 which zooms in on the green office chair from the reconstruction.

Figure 33: Distortions in reconstruction.

These distortions occurred during the angular motion of the robot. Consider Figure 34 which depicts the opposite side of the room. There was a minimal rotational motion from the robot while capturing the data in this section and it can be seen that the reconstructed point cloud is much more coherent than the other section with the chairs.

(a) Reconstructed pointcloud.

(b) Image of the section taken with a phone camera.

Figure 34: The opposite side of the room.

Another aspect that is observed to be affecting the quality in the real environment is the camera location. As observed in simulations it is imperative to acquire at least one complete leg, otherwise, it might lead to the replaced models not being grounded. The camera location on the Care-O-bot is very far up from the ground. Since the camera is high up, the robot has to be at a farther distance to capture the full extent of the object. It was noticed that the depth information of far-away objects has high noise content. This makes it hard to capture the legs of the chair especially when they are thin. Evidently, Figure 35 shows that the marked chairs are floating in the air as the robot failed to capture the thin legs. The only chair which is grounded is an office chair with a thick stem that supports the upper body which got properly registered. Another observation from the Figure is that all the floating chairs are far from the actual chairs which are small wooden chairs. This issue can be attributed to the label spreading caused by Mask R-CNN. As the legs are thin, depth information is noisy and the mask is inaccurate the segmented object cluster included the mislabeled background along with the object's point cloud. This resulted in obtaining a match that was far different from the actual object.

Adapting from the fact that the Care-O-bot needs to be at a far distance and

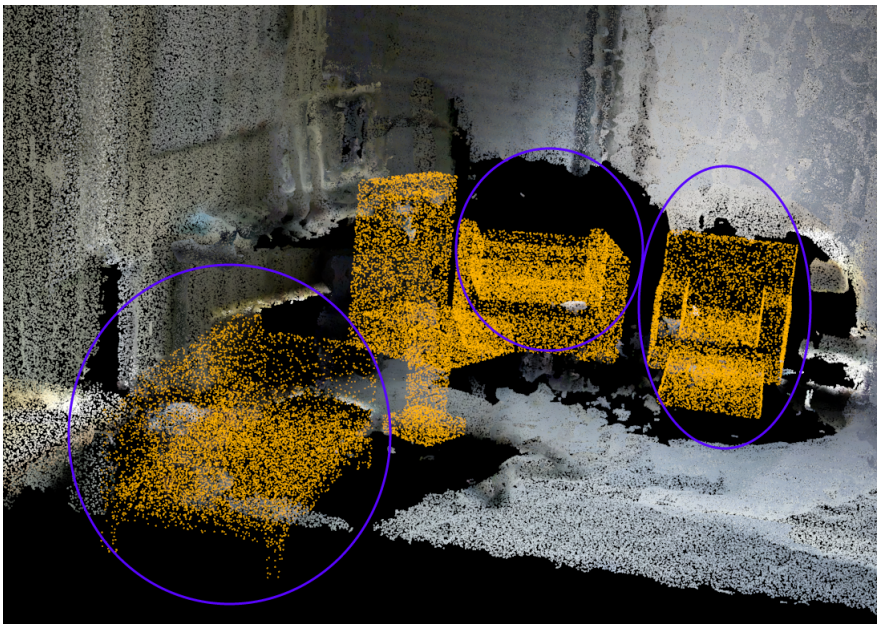


Figure 35: Zoomed-in version of figure 32b.

moved minimally, the same area was mapped again with the robot being stationary. Consider Figure 36, the first thing to notice is that the geometric reconstruction is much more coherent than before and secondly more number of chairs got replaced. The F_1-score is found to be 0.78 which is not a big improvement from the previous test even though more number of chairs were identified. This low improvement can be attributed to the presence of two false positives which were not present earlier.

The robot had a clear view of chairs 1 and 4 and hence a reasonable and grounded match was found for them. Chair 3 was farther away from chair 4 contrary to how it is perceived from the image, the legs were not registered due to the increased noise in depth information for far-away objects. As for chair 5, the majority of its structure was blocked by a chair. However, through the hole in the backrest of chair 4 and through the sides, part of the chair 5's structure was registered and hence a match was found for it. As the legs got completely occluded by chair 4 the replaced model is seen floating in the scene. The False positives are caused by chair 6. The wheeled base, backrest, and seat were registered whereas the stem connecting the seat with the base was not registered. Hence, it got clustered into two different sets resulting in a false positive as seen below chair 6. As for the other false-positives, it is a result of label spreading caused because of Mask R-CNN by which the part of the wall behind got mislabeled.

(a) Reconstructed area.

(b) Area after chairs are replaced.

Figure 36: Same area mapped with robot at standstill.

4.2.2 Care-O-bot in corridor

(a) Reconstructed corridor.

(b) Reconstructed corridor with chairs replaced.

Figure 37: Care-O-bot in corridor.

A corridor is a common place where an office robot may navigate to reach its final destination. When the corridors are narrow and there is no space for the robot

to circumnavigate the object in its path, the robot often gets halted till that block is cleared. Instead, it is helpful if the robot knows how to manipulate the said object to clear a path for itself.

4 chairs are laid out in a narrow corridor and the robot maps the area. As per the analysis in Section 4.2.1 the camera location and the velocity of the robot was observed to influence the quality of the pipeline output, hence, for this test, the robot was at a standstill with chairs at a reasonable distance from the robot. Figure 37 depicts the result of the pipeline. A F_1-score of 0.89 was calculated based on the result.

Analysis

The result observed was as expected; the first three chairs which the robot had a clear view of were matched to very similar models whereas the chair at the far most end was matched with a wrong model as the legs of the chair completely evaded the vision of the robot. The mask generated on chairs does not account for the hollow regions in them like the ones in the backrest of the first three chairs, thus the part of the wall got mislabeled to become a false positive.

Conclusion

The pipeline shows a level of performance similar to that of simulation. All the inherent shortcomings regarding the Mask R-CNN and the pace of the robot are reflected in the performance of the pipeline for the real environment as well. The effect of pace was seen to be more predominant than in the simulation as the sensor onboard the robot provided the image inputs at only half the rate than that of the simulation which introduced distortions in the geometric reconstruction due to lack of timely data that matches with the pace. Additionally, it was observed that the camera location and the noise in the depth information also influenced the final output.

4.3 Discussion

The experiments conducted in simulated and real environments gave a detailed insight into the capabilities and limitations of the pipeline. The main takeaway from the experiments was that the final output of the pipeline heavily relies on the input data. The raw input from the RGBD cameras undergoes transformations or modifications at all stages in the pipeline. The discrepancies and errors introduced at each stage accumulate and carry forward which affects the overall performance of the pipeline. Voxblox output is distorted when the input frequency is not matching with the pace of mapping, Mask R-CNN introduces label spreading and false detection ultimately resulting in false positives, shape filter is unsuccessful when encountering a non-chair object with a size comparable to a chair, Model matching provides a sub-optimal solution when a significant part of the object is completely missing, object replacement process removes parts of the background.

In terms of reliability, the pipeline will give consistent output for the same geometric and semantic geometric reconstruction with the same exact match found from the database and F_1-score for every run. However, the final output will vary when the pipeline is subjected to a different geometric and sematic-geometric reconstruction of the same exact environment. As data collected during each run mapping the same environment to build multiple pairs of reconstructions differ in terms of data density and errors introduced by different modules, the final output also reflects these differences.

The findings from the simulation and real data are conclusive towards establishing the limitations and capabilities of the pipeline in general. Keeping the limitations in mind it will be quite interesting to discuss how the pipeline would adapt when the database is expanded to other grounded objects, for example, types of tables. As tables are larger in size unlike chairs the false positives due to label spreading will have a lesser impact as the non-table objects get filtered out during the filtering process. As the tables have flat surfaces over the support structure the height of the robot will also impact the output depending upon the visibility of the tabletop resulting in situations where the matched table may not completely cover the entire area that the actual table occupies. For the cases where long tables are involved, it may get replaced by two shorter tables depending on whether the robot was able to register the complete extent of the tabletop without any missing regions in between.

Out of all the existing solutions used PCN method was the biggest disappointment. The method performed well on the original training data but completely fell apart when subjected to data from the simulation. The analysis shows clear signs of overfitting to the training data and the employed training methodology. Alternatively, it may also be because of the direct use of a trained model and may possibly work better if the network is trained from scratch with additional training data apart from the ShapeNet models. A similar explanation is also possible for Mask R-CNN as well, a network trained with an expanded database may provide a more accurate object mask reducing the label spreading.

To conclude, the detailed analysis of the experiments conducted managed to answer all the questions posed which address different aspects of the pipeline. The model matching method outperforms the PCN method, Mask R-CNN was identified to be the weakest module in the pipeline, the pipeline performance indicates a dependency on the robot speed while mapping as it impacts on the quality of the initial reconstructions, the pipeline translates well to noisy sensor data providing comparable performance to that in simulation.

5 Conclusion and future work

In this thesis, a pipeline to generate a 3D mesh representation of an environment with relevant objects replaced with similar synthetic models was designed and implemented. The pipeline consists of modules each with a specific task. The first half of the pipeline takes the camera inputs along with the pose information to generate a geometric and a semantic-geometric mesh representation of the environment. Voxblox and Kimera combined with Mask R-CNN, which are existing solutions, were used to generate and save the geometric and semantic-geometric mesh respectively. In the second half of the pipeline, the saved meshes are converted into a point cloud for more flexible processing. The semantic point cloud is broken down into individual object clusters and the clusters of chairs are filtered out. The final module attempts to infer the missing regions in the individual clusters and replace the completed version into the geometric point cloud. For this end module, two different approaches were tried. First, a deep learning based method, PCN which produces a complete point cloud when a partial point cloud is provided as input. Second, a novel approach to match the partial point clouds with a database of synthetic models.

The deep learning and model matching approaches were compared against each other based on execution time and the visual quality of the outputs in simulation. A Husky robot model in an office environment in Gazebo was used as the simulation setup. The PCN methods performed well on time but completely fell apart on generating quality outputs. Additionally, the performance of the overall pipeline was evaluated based on F_1-score and the variation in F_1-score for different speed settings for the robot. The pipeline performed reasonably well on low speed and suffered from quality loss for moderate and high speeds. The detailed analysis showed the major bottlenecks for the pipeline quality was due to the inaccurate and slow output frequency from the image segmentation method (Mask R-CNN). Furthermore, it was also observed that the model matching method requires data from at least one leg to identify a reasonable match for the partial input.

The pipeline was also tested in a real environment using a Care-O-bot. The results obtained were similar to that of simulation which showed dependency on the speed of navigation through the environment and the quality of the output of Mask R-CNN. Additionally, it was also observed that the camera location and the noisy depth information impacted the quality of the output.

For future work, the pipeline quality can be improved by replacing the current image segmentation method with a better one which is capable of running at a higher frequency and which provides better mask accuracy or retrain the existing network incorporating additional data which may result in better mask accuracy. The model database can be expanded to include more everyday objects which are grounded such as tables, waste bins, sofas, plant pots. The approach to remove objects from the current representation was to remove data around that object in a set radius which results in overall data loss. A better way can be explored to remove only the relevant objects without significant loss in background information.

Apart from improving the pipeline, the uses of the output of the pipeline can also be explored. A possible application would be to use the output to spawn a

robot in the generated environment and learn to manipulate specific objects for the purpose of 'Interactive Navigation'. The ultimate purpose of a semantic map is to represent the environment as accurately as possible to aid the robot to perform complex tasks where accuracy refers to how much information about the environment is being conveyed. The scene completion approach brings improvement to the conveyed geometric information. However, this approach can evolve to encode more information about the object such as affordances, contact points by utilizing more sophisticated databases that may not be existing at the moment. In essence, the approach can act as a channel to incorporate additional information into the scene to aid the robot.

References

[1] D. Goodwin. (Sep. 9, 2020). "The Evolution of Autonomous Mobile Robots - Technical Articles," [Online]. Available: https://control.com/technical-articles/the-evolution-of-autonomous-mobile-robots/ (visited on 11/03/2020).

[2] R. Barber, J. Crespo, C. Gómez, A. C. Hernámdez, and M. Galli, "Mobile Robot Navigation in Indoor Environments: Geometric, Topological, and Semantic Navigation," in *Applications of Mobile Robots*, ser. Mobile Robotics 26325195, vol. 1, IntechOpen, Nov. 5, 2018, p. 228, ISBN: 978-1-78985-755-9 978-1-78985-756-6. [Online]. Available: https://www.intechopen.com/books/applications-of-mobile-robots/mobile-robot-navigation-in-indoor-environments-geometric-topological-and-semantic-navigation (visited on 11/03/2020).

[3] J. Boal, Á. Sánchez-Miralles, and Á. Arranz, "Topological simultaneous localization and mapping: A survey," *Robotica*, vol. 32, pp. 803–821, Aug. 1, 2014. DOI: 10.1017/S0263574713001070.

[4] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016, ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2016.2624754. arXiv: 1606.05830. [Online]. Available: http://arxiv.org/abs/1606.05830 (visited on 11/03/2020).

[5] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010, pp. 797–803. DOI: 10.1109/IROS.2010.5654369.

[6] F. Xia, W. B. Shen, C. Li, P. Kasimbeg, M. E. Tchapmi, A. Toshev, R. Martín-Martín, and S. Savarese, "Interactive Gibson Benchmark: A Benchmark for Interactive Navigation in Cluttered Environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 713–720, Apr. 2020, ISSN: 2377-3766. DOI: 10.1109/LRA.2020.2965078.

[7] E. Ahmed, A. Saint, A. El Rahman Shabayek, K. Cherenkova, R. Das, G. Gusev, D. Aouada, and B. Ottersten, "A survey on deep learning advances on different 3D data representations," *arXiv e-prints*, arXiv:1808.01462, arXiv:1808.01462, Aug. 2018. arXiv: 1808.01462 [cs.CV].

[8] A. M. Hafiz and G. M. Bhat, "A survey on instance segmentation: State of the art," *International Journal of Multimedia Information Retrieval*, vol. 9, no. 3, pp. 171–189, Sep. 1, 2020, ISSN: 2192-662X. DOI: 10.1007/s13735-020-00195-x. [Online]. Available: https://doi.org/10.1007/s13735-020-00195-x (visited on 11/05/2020).

[9] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Gläser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–20, 2020, ISSN: 1558-0016. DOI: 10.1109/TITS.2020.2972974.

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, Curran Associates, Inc., 2015, pp. 91–99. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[12] L. Chen, A. Hermans, G. Papandreou, F. Schroff, P. Wang, and H. Adam, "MaskLab: Instance Segmentation by Refining Object Detection with Semantic and Direction Features," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 4013–4022. DOI: 10.1109/CVPR.2018.00422.

[13] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 8759–8768. DOI: 10.1109/CVPR.2018.00913.

[14] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun, "MegDet: A Large Mini-Batch Object Detector," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 6181–6189. DOI: 10.1109/CVPR.2018.00647.

[15] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, C. C. Loy, and D. Lin. (Apr. 9, 2019). "Hybrid Task Cascade for Instance Segmentation." arXiv: 1901.07518 [cs], [Online]. Available: http://arxiv.org/abs/1901.07518 (visited on 11/06/2020).

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., Cham: Springer International Publishing, 2016, pp. 21–37, ISBN: 978-3-319-46448-0.

[18] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-Time Instance Segmentation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019, pp. 9156–9165. DOI: 10.1109/ICCV.2019.00925.

[19] X. Chen, R. Girshick, K. He, and P. Dollar, "TensorMask: A Foundation for Dense Object Segmentation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South): IEEE, Oct. 2019, pp. 2061–2069, ISBN: 978-1-72814-803-8. DOI: 10.1109/ICCV.2019.00215. [Online]. Available: https://ieeexplore.ieee.org/document/9010024/ (visited on 11/06/2020).

[20] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, Oct. 2011, pp. 127–136. DOI: 10.1109/ISMAR.2011.6092378.

[21] K. Chen, Y.-K. Lai, and S.-M. Hu, "3D indoor scene modeling from RGB-D data: A survey," *Computational Visual Media*, vol. 1, no. 4, pp. 267–278, Dec. 1, 2015, ISSN: 2096-0662. DOI: 10.1007/s41095-015-0029-x. [Online]. Available: https://doi.org/10.1007/s41095-015-0029-x (visited on 11/09/2020).

[22] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, "Kintinuous: Spatially extended KinectFusion," in *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, Australia, Jul. 2012.

[23] F. Steinbrücker, J. Sturm, and D. Cremers, "Volumetric 3D mapping in real-time on a CPU," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2021–2028, Sep. 2014. DOI: 10.1109/ICRA.2014.6907127.

[24] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Trans. Graph.*, vol. 32, no. 6, Nov. 2013, ISSN: 0730-0301. DOI: 10.1145/2508363.2508374. [Online]. Available: https://doi.org/10.1145/2508363.2508374.

[25] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 1366–1373. DOI: 10.1109/IROS.2017.8202315.

[26] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "SemanticFusion: Dense 3D semantic mapping with convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4628–4635. DOI: 10.1109/ICRA.2017.7989538.

[27] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1697–1716, 2016. DOI: 10.1177/0278364916669237. eprint: https://doi.org/10.1177/0278364916669237. [Online]. Available: https://doi.org/10.1177/0278364916669237.

[28] M. Grinvald, F. Furrer, T. Novkovic, J. J. Chung, C. Cadena, R. Siegwart, and J. Nieto, "Volumetric Instance-Aware Semantic Mapping and 3D Object Discovery," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 3037–3044, Jul. 2019, ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2923960.

[29] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: An Open-Source Library for Real-Time Metric-Semantic Localization and Mapping," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 1689–1696. DOI: 10.1109/ICRA40945.2020.9196885.

[30] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "Laplacian mesh optimization," in *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*, ser. GRAPHITE '06, New York, NY, USA: Association for Computing Machinery, 2006, pp. 381–389, ISBN: 1-59593-564-9. DOI: 10.1145/1174429.1174494. [Online]. Available: https://doi.org/10.1145/1174429.1174494.

[31] W. Zhao, S. Gao, and H. Lin, "A robust hole-filling algorithm for triangular mesh," *The Visual Computer*, vol. 23, no. 12, pp. 987–997, Dec. 1, 2007, ISSN: 1432-2315. DOI: 10.1007/s00371-007-0167-y. [Online]. Available: https://doi.org/10.1007/s00371-007-0167-y.

[32] O. Sorkine and D. Cohen-Or, "Least-squares meshes," in *Proceedings Shape Modeling Applications, 2004.*, Jun. 2004, pp. 191–199. DOI: 10.1109/SMI.2004.1314506.

[33] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Symposium on Geometry Processing*, A. Sheffer and K. Polthier, Eds., The Eurographics Association, 2006, ISBN: 3-905673-24-X. DOI: 10.2312/SGP/SGP06/061-070.

[34] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Trans. Graph.*, vol. 32, no. 3, Jul. 2013, ISSN: 0730-0301. DOI: 10.1145/2487228.2487237. [Online]. Available: https://doi.org/10.1145/2487228.2487237.

[35] I. Sipiran, R. Gregor, and T. Schreck, "Approximate symmetry detection in partial 3D meshes," *Computer Graphics Forum*, vol. 33, no. 7, pp. 131–140, 2014. DOI: 10.1111/cgf.12481. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12481. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12481.

[36] N. J. Mitra, L. J. Guibas, and M. Pauly, "Partial and approximate symmetry detection for 3D geometry," in *ACM SIGGRAPH 2006 Papers*, ser. SIGGRAPH '06, New York, NY, USA: Association for Computing Machinery, 2006, pp. 560–568, ISBN: 1-59593-364-6. DOI: 10.1145/1179352.1141924. [Online]. Available: https://doi.org/10.1145/1179352.1141924.

[37] A. C. Pablo Speciale Martin R. Oswald and M. Pollefeys, "A symmetry prior for convex variational 3D reconstruction," in *European Conference on Computer Vision (ECCV)*, 2016.

[38] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L. J. Guibas, "Discovering structural regularity in 3D geometry," *ACM Trans. Graph.*, vol. 27, no. 3, p. 43, 2008. DOI: 10.1145/1360612.1360642. [Online]. Available: https://doi.org/10.1145/1360612.1360642.

[39] A. Dai, C. R. Qi, and M. Nießner. (Apr. 11, 2017). "Shape Completion using 3D-Encoder-Predictor CNNs and Shape Synthesis." arXiv: 1612.00101 [cs], [Online]. Available: http://arxiv.org/abs/1612.00101 (visited on 11/08/2020).

[40] A. Sharma, O. Grau, and M. Fritz, "VConv-DAE: Deep Volumetric Shape Learning Without Object Labels," *ECCV Workshops*, 2016. DOI: 10.1007/978-3-319-49409-8_20.

[41] X. Han, Z. Li, H. Haibin, E. Kalogerakis, and Y. Yu, "High-Resolution Shape Completion Using Deep Neural Networks for Global Structure and Local Geometry Inference," Oct. 24, 2017. DOI: 10.1109/ICCV.2017.19.

[42] M. Sarmad, H. J. Lee, and Y. M. Kim, "RL-GAN-Net: A reinforcement learning agent controlled GAN network for real-time point cloud shape completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[43] O. Litany, A. Bronstein, M. Bronstein, and A. Makadia, "Deformable Shape Completion with Graph Convolutional Autoencoders," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 1886–1895. DOI: 10.1109/CVPR.2018.00202.

[44] S. A. Bello, S. Yu, C. Wang, J. M. Adam, and J. Li, "Review: Deep learning on 3D point clouds," *Remote Sensing*, vol. 12, no. 11, 2020, ISSN: 2072-4292. DOI: 10.3390/rs12111729. [Online]. Available: https://www.mdpi.com/2072-4292/12/11/1729.

[45] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 77–85. DOI: 10.1109/CVPR.2017.16.

[46] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "PCN: Point Completion Network," in *2018 International Conference on 3D Vision (3DV)*, Sep. 2018, pp. 728–737. DOI: 10.1109/3DV.2018.00088.

[47] Z. Huang, Y. Yu, J. Xu, F. Ni, and X. Le, "PF-Net: Point Fractal Network for 3D Point Cloud Completion," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 7659–7667. DOI: 10.1109/CVPR42600.2020.00768.

[48] X. Chen, B. Chen, and N. J. Mitra, "Unpaired point cloud completion on real scans using adversarial training," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020. [Online]. Available: https://openreview.net/forum?id=HkgrZ0EYwB.

[49] L. P. Tchapmi, V. Kosaraju, H. Rezatofighi, I. Reid, and S. Savarese, "TopNet: Structural point cloud decoder," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[50] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 3212–3217. DOI: 10.1109/ROBOT.2009.5152473.

[51] B. Drost and S. Ilic, "3D Object Detection and Localization Using Multimodal Point Pair Features," in *Visualization Transmission 2012 Second International Conference on 3D Imaging, Modeling, Processing*, Oct. 2012, pp. 9–16. DOI: 10.1109/3DIMPVT.2012.53.

[52] A. Avetisyan, M. Dahnert, A. Dai, M. Savva, A. X. Chang, and M. Niessner, "Scan2CAD: Learning CAD model alignment in RGB-D scans," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[53] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 2432–2443. DOI: 10.1109/CVPR.2017.261.

[54] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An information-rich 3D model repository," Stanford University — Princeton University — Toyota Technological Institute at Chicago, arXiv:1512.03012 [cs.GR], 2015.

[55] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: An open-source Robot Operating System," p. 6, 2009.

[56] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '87, New York, NY, USA: Association for Computing Machinery, 1987, pp. 163–169, ISBN: 0-89791-227-6. DOI: 10.1145/37401.37422. [Online]. Available: https://doi.org/10.1145/37401.37422.

[57] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2014, pp. 740–755, ISBN: 978-3-319-10602-1. DOI: 10.1007/978-3-319-10602-1_48.

[58] I. Culjak, D. Abram, T. Pribanic, H. Dzapo, and M. Cifrek, "A brief introduction to OpenCV," in *2012 Proceedings of the 35th International Convention MIPRO*, May 2012, pp. 1725–1730.

[59] Q.-Y. Zhou, J. Park, and V. Koltun. (2018). "Open3D: A modern library for 3D data processing." arXiv: 1801.09847.

[60] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9–13, 2011.

[61] Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 206–215. DOI: 10.1109/CVPR.2018.00029.

[62] K. M. Jatavallabhula, E. Smith, J.-F. Lafleche, C. F. Tsang, A. Rozantsev, W. Chen, T. Xiang, R. Lebaredian, and S. Fidler. (Nov. 13, 2019). "Kaolin: A PyTorch Library for Accelerating 3D Deep Learning Research." arXiv: 1911.05063 [cs], [Online]. Available: http://arxiv.org/abs/1911.05063 (visited on 11/16/2020).

[63] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 3212–3217. DOI: 10.1109/ROBOT.2009.5152473.

[64] A. Rasouli and J. K. Tsotsos. (Feb. 14, 2017). "The Effect of Color Space Selection on Detectability and Discriminability of Colored Objects." arXiv: 1702.05421 [cs], [Online]. Available: http://arxiv.org/abs/1702.05421 (visited on 11/18/2020).

[65] (). "Gazebo," [Online]. Available: http://gazebosim.org/ (visited on 11/18/2020).

[66] R. Kittmann, T. Fröhlich, J. Schäfer, U. Reiser, F. Weißhardt, and A. Haug, "Let me introduce myself: I am care-o-bot 4, a gentleman robot," in *Mensch Und Computer 2015 - Proceedings, Stuttgart, Germany, September 6-9, 2015*, S. Diefenbach, N. Henze, and M. Pielot, Eds., De Gruyter Oldenbourg, 2015, pp. 223–232. [Online]. Available: https://dl.gi.de/20.500.12116/7892.